GIGI™

GIGI/ReGIS Handbook

digital
DIGITAL EQUIPMENT CORPORATION

June 1981

GIGI™/ReGIS Handbook

Order Number AA-K336A-TK

This manual introduces GIGI's hardware features and graphics capabilities. Also, the manual provides reference information on ReGIS (the Remote Graphics Instruction Set), GIGI's set-ups, and GIGI-supported escape sequences. The manual also provides information on how to program using GIGI features.

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754.

DIGITAL EQUIPMENT CORPORATION

First Printing June 1981

The postage paid Reader's Comments Form on the last page of this document requests your critical evaluation to assist us in preparing future documentation.

CONTENTS

Appendices and Glossary

PREFACE

MANUAL OBJECTIVES

This manual describes the user-oriented features of the GIGI terminal:

- Terminal set-up features, that is, those features that allow you to adapt GIGI to many operating environments.
- ReGIS (the Remote Graphics Instruction Set), the graphics language you use to display graphics on the GIGI monitor screen.
- Programming information and examples for programming the terminal and using GIGI graphics.

INTENDED AUDIENCE

This manual is for programmers responsible for programming terminal features and for programming graphics.

STRUCTURE OF THE DOCUMENT

The manual has three parts:

Part 1: Getting Started provides an overview of GIGI's components and architecture. This part also describes how to set up the terminal to work properly in different operating situations; lists all the terminal features that you can control from the keyboard.

Part 2: GIGI Graphics introduces graphics concepts and the ReGIS graphics language. This part of the manual also contains reference information on all of the ReGIS commands and their syntax.

Part 3: Programming GIGI describes how to program the programmable keypad, use locator mode, and program using escape sequences. Also, this part provides examples of graphics programming in high-level languages such as BASIC and PASCAL.

ASSOCIATED DOCUMENTS

GIGI BASIC Manual *Order Number AA-K335A-TK*

Hardware Documentation
VK100 Terminal Installation and Owners Manual
Order Number EK-VK100-IN
LA34RO User's Guide *Order Number EK-L34RO-UG*

Software Documentation
GIGI provides software that can run on many DIGITAL operating systems, for example, VAX/VMS, RSTS/E, and TOPS-20. Ask your DIGITAL sales representative for information about these software packages.

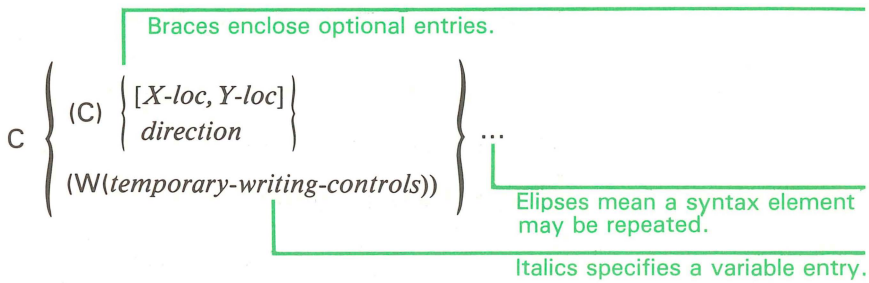
CONVENTIONS USED IN THIS DOCUMENT

Braces: indicate optional entries; choose one or more of the elements within the braces. All parentheses and brackets that appear within the command syntax must be coded as shown.

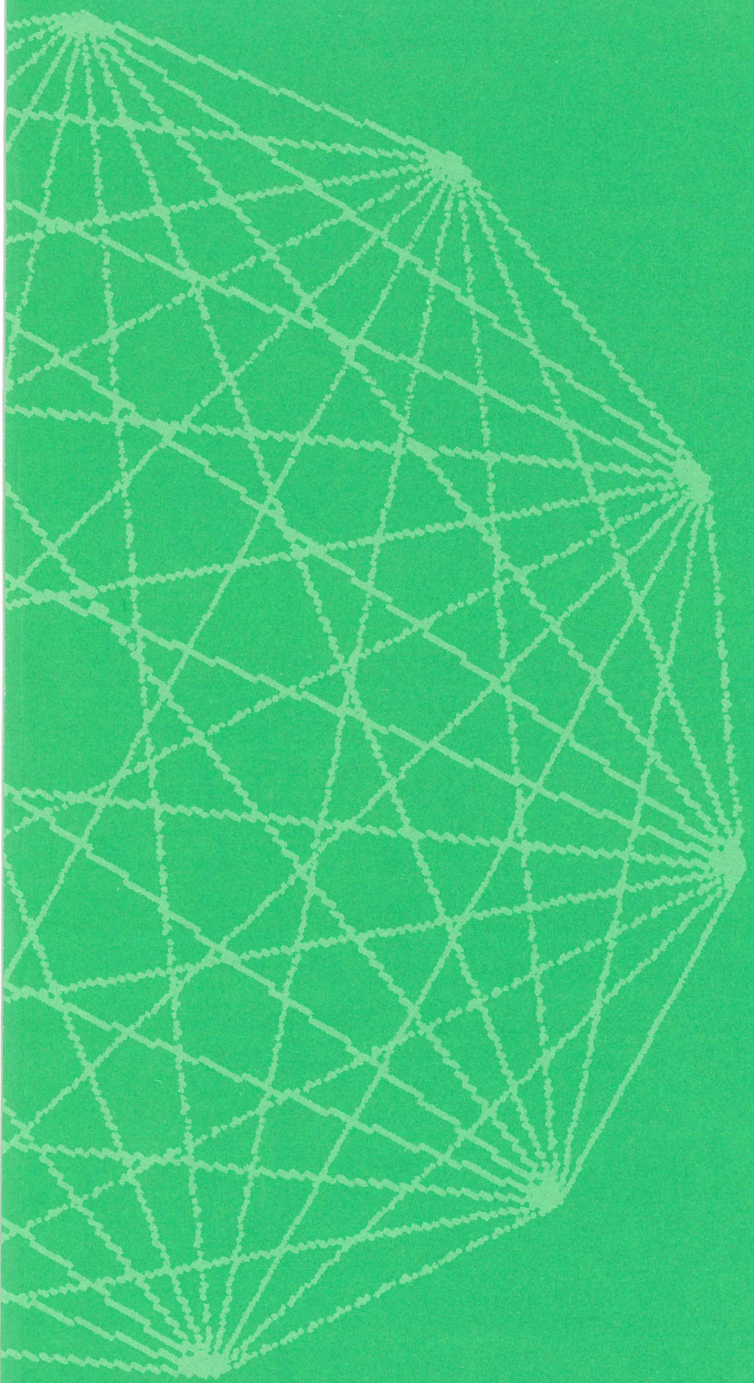
Elipses: indicate that the preceding command element or elements may be repeated.

Italics: indicates a variable entry for a ReGIS command option, both in syntax specifications and in detailed descriptions of options.

The following sample syntax specification illustrates these conventions:



1



Getting Started

Chapter 1 introduces GIGI's components and describes GIGI's terminal architecture.

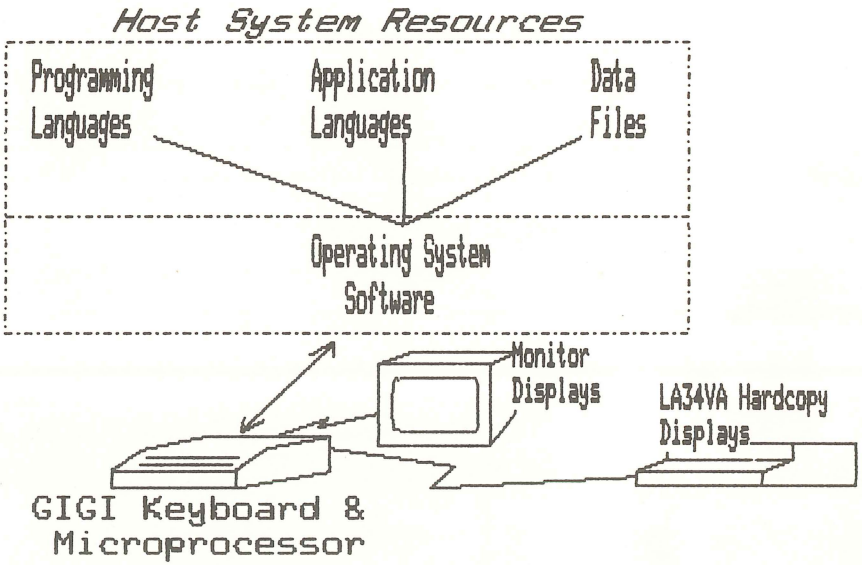
Chapter 2 contains information on the terminal set-ups, features which allow you to adapt GIGI to many operating environments.

1 INTRODUCTION TO GIGI

GIGI (the General Imaging Generator and Interpreter) is a terminal designed for use as terminal subsystem connected to a host computer. GIGI provides local (in-terminal) processing using a microprocessor. The microprocessor supports two interpreters: a ReGIS graphics interpreter and the GIGI BASIC language interpreter. ReGIS (the Remote Graphics Instruction Set) is a graphics language; GIGI BASIC is a BASIC language designed to make use of GIGI's unique graphic capabilities.

GIGI is a separate keyboard which requires a user-supplied monitor for displaying screen images. Either black-and-white or color (RGB) monitors can be used. Also, GIGI can display images on an optional LA34VA Graphics Printer, which attaches directly to the terminal. Figure 1-1 shows GIGI's component parts.

Figure 1-1.
GIGI's components



The document *GIGI Terminal Installation and Owner's Manual*, Order Number EK-VK100-IN, describes GIGI's terminal hardware components in detail and tells you how to operate the terminal; the document *DECwriter IV Graphics Printer User Guide*, Order Number EK-L34RO-UG, describes the LA34VA graphics printer and how to operate it.

TERMINAL OPERATING MODES

GIGI can be used as a text terminal and as a graphics terminal. As a text terminal, GIGI can display the following:

- Text entered from the keyboard.
- Operating system-generated information, such as command language prompts.
- Output from host programs (for example, listings).
- Interactive dialogue generated by host programs.
- Output from the local BASIC interpreter.

As a graphics terminal, GIGI provides an interface to the ReGIS graphics interpreter. Using this interface (an escape sequence), GIGI can display pictures (including lines, curves, and text) on the video monitor.

GIGI TEXT MODE

In text mode (GIGI's power up mode), GIGI can be used as an interactive video display terminal. GIGI can be used either locally with the BASIC interpreter (set-up BA1) or as an interactive video display terminal attached to a host computer system (setups LL1 and BA0). Local interactive processing is controlled by the BASIC interpreter; online interactive processing is controlled by host resources.

In text mode, GIGI displays the ASCII characters listed in Table 1-1.

Table 1-1. ASCII Characters GIGI Displays in Text Mode

ASCII Code	Character	ASCII Code	Character	ASCII Code	Character
000	NUL	043	+	086	V
001	SOH	044	,	087	W
002	STX	045	-	088	X
003	ETX	046	.	089	Y
004	EOT	047	/	090	Z
005	ENQ	048	0	091	[
006	ACK	049	1	092	×
007	BEL	050	2	093	]
008	BS	051	3	094	^
009	HT	052	4	095	_

ASCII Code	Character	ASCII Code	Character	ASCII Code	Character
010	LF	053	5	096	'
011	VT	054	6	097	a
012	FF	055	7	098	b
013	CR	056	8	099	c
014	SO	057	9	100	d
015	SI	058	:	101	e
016	DLE	059	;	102	f
017	DC1	060	<	103	g
018	DC2	061	=	104	h
019	DC3	062	>	105	i
020	DC4	063	?	106	j
021	NAK	064	@	107	k
022	SYN	065	A	108	l
023	ETB	066	B	109	m
024	CAN	067	C	110	n
025	EM	068	D	111	o
026	SUB	069	E	112	p
027	ESCAPE	070	F	113	q
028	FS	071	G	114	r
029	GS	072	H	115	s
030	RS	073	I	116	t
031	US	074	J	117	u
032	SPACE	075	K	118	v
033	!	076	L	119	w
034	"	077	M	120	x
035	#	078	N	121	y
036	$	079	O	122	z
037	%	080	P	123	{
038	&	081	Q	124	|
039	'	082	R	125	}
040	(	083	S	126	~
041	)	084	T	127	DEL
042	*	085	U		

ASCII codes are in decimal. LF = Line Feed, FF = Form Feed, CR = Carriage Return, DEL = Delete BS = BackSpace

GIGI GRAPHICS MODE

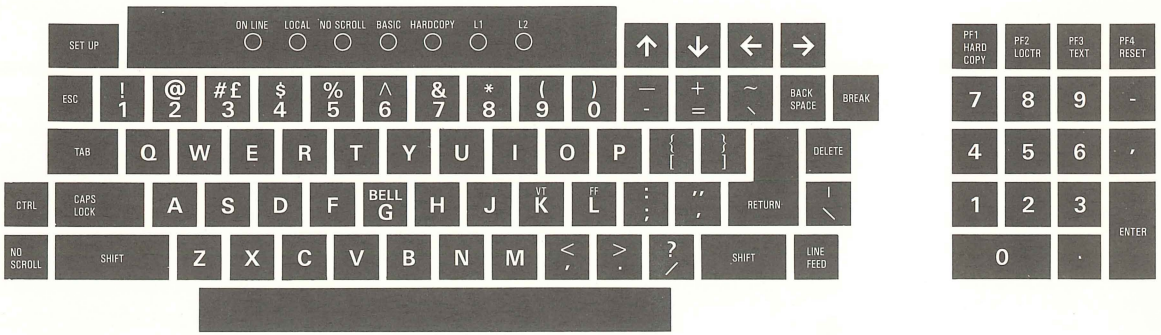
GIGI enters graphics mode when it receives a special escape sequence, <ESC>Pp (escape character followed by uppercase P and lowercase p). GIGI uses this escape sequence to enable graphics mode. Once in graphics mode, all characters transmitted to the terminal are interpreted as ReGIS graphics commands. GIGI directs the commands to the interpreter, which, in turn, generates the lines, curves, circles, and text which are displayed as pictures on the monitor screen.

You can enter graphics mode directly from the terminal, from local BASIC, and by means of host programs. Chapter 3, *Introduction to GIGI/ReGIS Graphics* introduces graphics concepts and provides examples of how to generate pictures. Chapter 4, *Executing ReGIS Commands*, describes how GIGI interprets and executes graphics commands.

GIGI'S KEYBOARD

The GIGI keyboard consists of the main keypad and the auxiliary keypad. Figure 1-2 shows the location on the keyboard of both keypads.

Figure 1-2.
GIGI Keyboard Layout



The Main Keypad

The main keypad consists of the ASCII keyboard characters shown in Figure 1-2. Also on the main keypad are the SET-UP key and the ESC and CTRL keys. The SET-UP key places the terminal in set-up mode. In set-up mode, you can set terminal features to fit your operating environment. Chapter 2, *GIGI SET-UPS*, describes GIGI set-up parameters and how to use them.

The ESC and CTRL keys transmit special character code sequences that can be interpreted by either a host operating system or GIGI or both. Part III, *Programming GIGI*, describes the functions of and how to program using ESC and CTRL sequences.

Main Keypad Arrow Keys

The arrow keys on the main keypad may be programmed to move the graphics cursor and current location. These keys transmit escape sequences which can be programmed and which can be controlled by means of set-ups. Part III, *Programming GIGI*, describes how to program and use the main keypad arrow keys.

The Auxiliary Keypad

The auxiliary keypad consists of 18 keys: four program function (PF) keys and 14 additional keys. This keypad operates in three modes: numeric, application, and programmed.

In both numeric and application modes, the four program function keys and the four cursor keys on the main keypad transmit standard escape sequences (listed in Part III, *Programming GIGI*). The remaining 14 keys transmit the values that appear on the keys, for example, the 5 key transmits the value 5. In application mode, these keys transmit standard escape sequences.

In programmed mode, all 18 keypad keys and the four cursor keys on the main keypad may be programmed to transmit sequences you define to suit your own programming needs. Part III, *Programming GIGI*, describes how to program the auxiliary keypad to fit unique applications.

GIGI'S PROGRAM FUNCTION KEYS

When used in combination with the SHIFT key, GIGI's Program Function keys (PF1, PF2, PF3, and PF4) perform special functions described in the following sections. These keys are located on the top row of the auxiliary keypad and are highlighted in Figure 1-1, above.

Printing the Screen Contents: SHIFT/PF1

When you press the SHIFT key and the PF1 key at the same time, GIGI prints the contents of the monitor screen on the LA34VA printer.

Entering Locator Mode: SHIFT/PF2

When you press the SHIFT key and the PF2 key at the same time, GIGI enters locator mode. In locator mode, GIGI displays a cross-hair cursor which you can move around the screen to the address (that is, the location) at which the graphics cursor is pointing. For information on how to use locator mode, refer to Part 3, *Programming GIGI.*

Returning to Text Mode From Graphics Mode: SHIFT/PF3

When you press the SHIFT key and the PF3 key at the same time, GIGI stops interpreting ReGIS commands, exits from graphics mode, and returns GIGI to text mode.

Setting GIGI Set-ups to Power-up Values: SHIFT/PF4

When you press the SHIFT key and the PF4 key at the same time, GIGI resets ReGIS to its initial (power-up) values and clears the screen. SHIFT/PF4 does not erase character sets, macrographs, or BASIC programs. SHIFT/PF4 stops GIGI BASIC processing.

MONITORS

GIGI can be attached to either a black-and-white or color monitor.

See your DEC sales representative for details on which monitors are appropriate for use with GIGI. Most types of black-and-white monitor and color monitors with RGB connectors can be used with GIGI.

Refer to the *GIGI Terminal Installation and Owner's Manual* for information on how to connect your GIGI to an appropriate monitor.

THE DECWRITER IV GRAPHICS PRINTER

The DECwriter IV is a dot matrix printer which can be attached to GIGI and used to print the contents of the screen, whether the screen contains simple text (for example, program output) or complex drawings. Refer to the *GIGI Terminal Installation and Owner's Manual* for information on how to connect the DECwriter IV Graphics Printer to GIGI and to the *DECWRITER IV Graphics Printer User Guide* for information on how to use the device with GIGI.

GRAPHICS TABLETS

You can attach an optional (user-supplied) graphics tablet to the hardcopy connector to provide input to GIGI in locator mode. The graphics tablet allows you to send a screen location to the host by means of a stylus or other device. Consult your DIGITAL sales representative for information on the selection and use of a graphics tablet with GIGI.

TERMINAL ARCHITECTURE

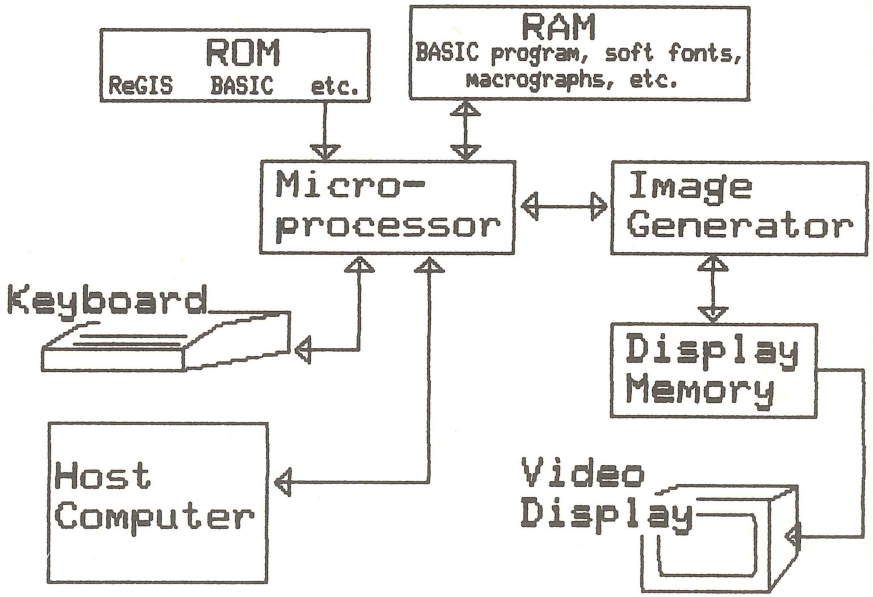
GIGI's terminal architecture is based on a single microprocessor, which interprets escape sequences, ReGIS commands, and GIGI BASIC commands and statements. This microprocessor controls most terminal functions by means of escape sequences. An escape sequence is a sequence of characters, preceded by the <ESC> character, which provide a specific control function for a given device. In this case the device is GIGI.

In general, escape sequences direct GIGI to perform particular functions. For example, GIGI interprets escape sequences that move the text cursor, erase a line of text, and set terminal operating characteristics such as keyclick and reverse video. All of the functions controlled by escape sequences are described in Chapter 16, *Programming with Escape Sequences*.

GIGI is designed to intercept escape sequences transmitted from either a host computer system (text mode) or directly from the keyboard (local mode). Chapter 5, *Executing ReGIS Commands*, describes how to use escape sequences and device control strings in more detail; Part III, *Programming GIGI*, describes the functions you can direct GIGI to perform using these sequences.

Figure 1-3 shows the relationship of the GIGI microprocessor to the GIGI terminal system.

Figure 1-3.
Relationship of GIGI Microprocessor to the GIGI Terminal System.



The Communications Interface and the Microprocessor

GIGI examines every character transmitted either from the keyboard when in local mode or from the host when GIGI is on-line. GIGI's microprocessor handles this processing and, depending on which character sequence is received, performs an operation.

The Image Generator and Display Memory

If the escape sequence is <ESC>Pp, GIGI interprets all characters following that sequence (until the next <ESC> character) as ReGIS graphics commands. In this case, the image generator is used to create an image based on the interpretation of ReGIS commands.

This image is maintained in display memory. Display memory is a special portion of GIGI's memory used to store the most-recently interpreted graphic image. Display memory is refreshed every 50th or 60th of a second, depending on how you set the power frequency.

ReGIS: A Generalized Graphics Language

GIGI's graphics interpreter accepts commands coded in ReGIS, the Remote Graphics Instruction Set. ReGIS is an easy-to-use language which provides commands for creating and manipulating color graphics and text images. For example, you can draw a picture of an apple in green and then label the apple in red. Part II, *GIGI Graphics*, describes the language and how to use it.

GIGI BASIC

The BASIC language interpreter operates on statements coded in the GIGI BASIC language. The GIGI BASIC interpreter processes programs locally and, by means of BASIC statements, uses the ReGIS interpreter to generate graphics on the video monitor screen. This language and how to use it are described in the *GIGI/BASIC Manual*.

PROGRAM WORKING STORAGE

GIGI provides approximately 8K bytes of storage for GIGI BASIC programs and data and approximately 2K bytes of storage for the programmable keypad sequences and macrographs. (Macrographs are graphics commands which you can store in GIGI memory and execute by means of a ReGIS command.)

2 GIGI SET-UPS

GIGI provides terminal control features called set-ups which make it easy to adapt GIGI to many operational situations. For example, set-ups allow you to use GIGI either as a VT52-type terminal or as an ANSI terminal.

Set-up mode is the terminal mode in which you can set GIGI to fit both your operating system environment and your work-station environment. For example, you can set the send/receive transmission rates to fit the needs of your host operating system; likewise you can set the keyboard keys to click or not, repeat or not, and so forth.

Set-ups are interactive, that is, you can see current set-up settings and easily change them directly from the GIGI keyboard. Figure 2-1 shows the appearance of the monitor screen when GIGI is in set-up mode.

The advantage of interactive set-ups is that they do not require elaborate procedures for setting terminal operating procedures such as send/receive transmission rates. The following sections describe how to set GIGI set-up parameters, list the set-ups by function, and describe each of the set-ups in detail.

ENTERING GIGI SET-UP MODE: THE SET-UP KEY

Enter set-up mode by pressing the SET-UP key on the upper-left corner of the main keypad. When you press the SET-UP key, GIGI scrolls the uppermost portion of the screen down into the video monitor screen to display the set-up window.

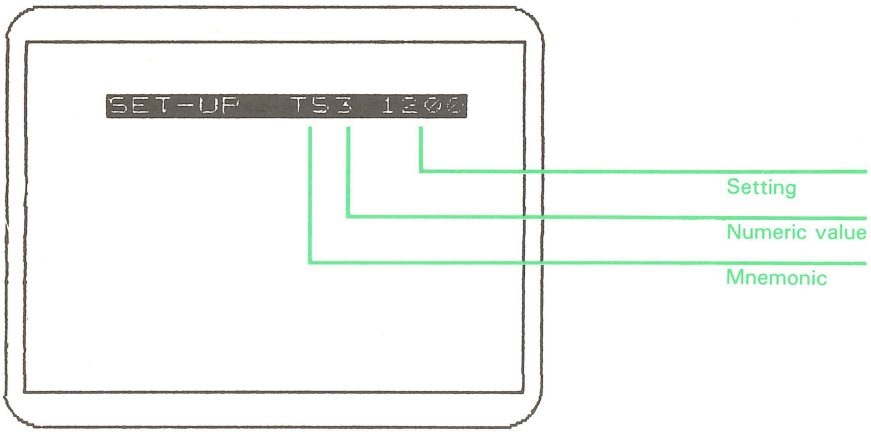


Figure 2-1.
Monitor Screen in GIGI Set-up Mode

In the sample screen in Figure 2-1, the first two characters are the mnemonic for the set-up parameter, in this case TS is the mnemonic for Transmit Speed. The mnemonic is followed by a numeric value which specifies the setting for the keyword; in the example, TS3 sets a transmission speed of 1200 baud. The value following the numeric keyword setting, 1200 in the example, shows the meaning of the numeric setting.

Set-ups fall into the following functional categories:

- Terminal control mode
- Keyboard control
- Screen control
- Special transmission modes
- Character control
- Communications control
- I/O and terminal hardware control
- GIGI BASIC control

Table 2-1 is a functionally organized list of the terminal features you can set by means of set-up parameters. Also, the table shows the parameters, their mnemonic values and meanings, and a brief description of the function the parameter performs.

Table 2-1. GIGI Set-up Categories by Function

Function	Parameter Mnemonic	Parameter Descriptions
Terminal Operating Mode	TM0 VT52 TM1 ANSI	Sets the escape sequence processing to VT52 or ANSI.
Keyboard Control	MB0 Off MB1 On	Enables or disables the margin bell.
	KC0 Off KC1 On	Enables or disables audible keyclick.
	KR0 Off KR1 On	Enables or disables autorepeat.
Screen Control	SM0 Off SM1 Jump SM2 Smth SM3 Wrap	Sets scrolling characteristics.
	RV0 Off RV1 On	Enables or disables reverse video.
	HM0-9	Specifies horizontal margins for screen image.
	HP0-9	Sets horizontal position of screen image.
	IL0 Off IL1 On	Enables or disables interlace.
	PF0 60Hz PF1 50Hz	Specifies the power frequency as either 50 or 60 Hz.
	VM0-9	Specifies vertical margins for screen image.
	VC0 Off VC1 Text VC2 Grph VC3 Both	Controls cursor visibility.
	AW0 Off AW1 On	Specifies whether to automatically generate a carriage return/line feed at the end of a line on the screen.

continued next page

Function	Parameter Mnemonic		Parameter Descriptions
Special Transmission Modes	LE0	Off	Enables or disables local echo of characters.
	LE1	On	
	NL0	Off	Specifies either new line (<CR><LF>) or no new line (<CR> only).
	NL1	On	
	SC0	Off	Specifies single character or line transmission.
	SC1	On	
Character Control	EM0	Off	Sets normal or double width (expanded) characters.
	EM1	On	
	OS0	Off	Enables or disables the character overstrike feature.
	OS1	On	
	TD0	Norm	Sets text mode control character display characteristics.
	TD1	Text	
	TD2	Ctrl	
	GD0	Norm	Sets the graphics debug display characteristics.
	GD1	Text	
	GD2	Top	
	GD3	Bot	
	GP0	Off	Enables or disables GIGI to interpret the graphics prefix character.
	GP1	1 = "!"	
	UK0	U.S.	Specifies the US or UK character set.
	UK1	U.K.	
Communication Controls	TS0	110	Set the speed at which GIGI transmits data.
	TS1	300	
	TS2	600	
	TS3	1200	
	TS4	2400	
	TS5	4800	
	TS6	9600	
	TS7	19.2	
	RS0	110	Set the speed at which GIGI receives data.
	RS1	300	
	RS2	600	

continued next page

Function	Parameter Mnemonic		Parameter Descriptions
Communication Controls continued	RS3	1200	
	RS4	2400	
	RS5	4800	
	RS6	9600	
	RS7	19.2	
	LL0	Locl	Specifies whether GIGI runs offline or under control of the host computer (online).
	LL1	Onln	
	PE0	Off	Specifies whether even, odd, or no parity is transmitted to and from the terminal.
	PE1	Even	
		Odd	
	XO0	Off	Specifies whether to use XON and XOFF synchronization characters.
	XO1	On	
	CI0	EIA	Specifies EIA or 20mA line interface.
	CI1	20mA	
	HS0	110	Sets transmission rate for hardcopy communications port.
	HS1	300	
	HS2	600	
	HS3	1200	
	HS4	2400	
	HS5	4800	
	HS6	9600	
	HS7	19.2	
I/O and Terminal Hardware Control	TL0	Off	Enables or disables the graphics tablet locator mode.
	TL1	On	
	AH0	Off	Specifies automatic activation of the hardcopy device.
	AH1	On	
	ST0	Clr	Invoke device self test functions.
	ST1	PwUp	
	ST2	ExCm	
	ST3	HcCm	
	ST4	Dspl	
	ST5	CBar	
	ST9	Rept	

continued next page

Function	Parameter Mnemonic		Parameter Descriptions
Programming Controls	CK0 CK1	Norm Appl	Sets DEC-defined escape sequences for cursor (arrow) keys.
	KP0 KP1	Norm Appl	Sets DEC-defined escape sequences for auxiliary keypad keys.
	PK0 PK1	Off On	Enables user-defined strings set for auxiliary keypad and cursor keys.
	TM0 TM1	VT52 ANSI	Enables auxiliary keypad for VT52 or ANSI control sequences.
BASIC Control	BA0 BA1 BA2	Off Locl Host	Enables or disables GIGI BASIC.

SETTING VALUES FOR SET-UP PARAMETERS

You can change the values of all of GIGI's set-up parameters either directly from the keyboard or by program controls. Certain set-up parameters can be changed manually using the switch-pack at the rear of the terminal. This section describes how to set set-ups directly from the keyboard and summarizes the settings of the switch-pack. Programming set-ups is described in Part III, *Programming With GIGI.*

Set-up Parameter Default Values

Each set-up parameter has a value associated with it when GIGI is turned on. These values remain as default values until you explicitly reset them. To determine the current set-up parameter values, press the SET-UP key to enter set-up mode and follow the procedures described below.

When power is turned off, set-ups which you have modified return to their power-up values.

Setting Set-up Parameter Values from the Keyboard

Set-up parameters have a fixed display sequence. When GIGI powers up or when you press SHIFT/RESET, the first parameter displayed in the set-up window is always the transmission speed parameter. You can use several methods (described below) to cycle through the parameter sequence and manipulate the set-up parameter values.

Determining Current Parameter Settings

Once the terminal is in set-up mode (GIGI is displaying the set-up window) you can determine current parameter settings using any of the following three methods:

- Press the RETURN or ENTER key each time you wish to advance to another set-up parameter.
- Press the right and left arrow keys on the main keypad to cycle forward and backward through the set-up parameters.
- Using the main keypad keys, enter the parameter mnemonic listed in Table 2-1 to look at a specific set-up parameter.

Changing Set-up Values

The parameter value you see in the set-up window is the value GIGI uses. As you cycle through the set-up parameters, you can either leave the parameter at the current value or change the value in one of the following ways.

Use the space bar to cycle through the allowable values for that parameter. Each time you press the space bar, GIGI sets the value of the set-up parameter to the next possible value.

Also, you can change the value of any set-up parameter by entering the set-up mnemonic and an appropriate numeric value (for example, KP1) from the main keypad. If you enter an invalid number for the parameter, GIGI ignores it.

EXITING FROM SETUP MODE

Once you have reset parameters appropriate to your needs, exit from set-up mode by pressing the SET-UP key.

Using Switches to Set Set-up Parameter Values

Using the eight switches (labelled 1-8) at the back panel of the keyboard, you can set power-up values for the following set-up features:

Switch	Set-up Feature
1	Power Frequency
2	Communications Interface
3	Default Character Set
4 and 5	Parity Enable
6, 7, and 8	Baud Rate for Transmit and Receive Speeds

Refer to the *GIGI Terminal Installation and Owner's Manual* for more information on these set-up features.

SET-UP PARAMETER DESCRIPTIONS

This section lists parameters alphabetically for quick reference. The alphabetical list contains detailed information on individual parameter specifications.

AH		**Auto-Hardcopy:** Enables or disables automatic printing of screen display each time GIGI refills the screen.
AH0	Off	Disables auto-hardcopy; screen is printed when either a ReGIS hardcopy command or a hardcopy escape sequence is received. AH0 is the default setting.
AH1	On	Enables auto-hardcopy; screen is printed automatically each time the screen fills.
AW		**Auto-wraparound:** Enables or disables the auto-wraparound feature.
AW0	Off	Disable autowrap; entering text in the right-most column leaves the text cursor in the right-most column and only the last character entered is visible.

AW1	On	Enable autowrap; characters entered beyond the right-most column wrap to the next line. Entering text in the right-most column leaves the text cursor in the right-most column. Subsequent text characters received are written on the next line until you enter a carriage return or line feed to the terminal. AW1 is the default setting.
BA		**GIGI BASIC:** Enables or disables GIGI BASIC; selects GIGI BASIC's operating mode.
BA0	Off	Disable GIGI BASIC. When GIGI BASIC is disabled, GIGI's functions are controlled by communicating with the host system. BA0 is the default setting.
BA1	Locl	Enable GIGI BASIC and process commands entered directly from the keyboard. GIGI directs BASIC output to the screen.
BA2	Host	Enable GIGI BASIC; BASIC input and output are controlled via the host system. In general, BA2 is used when the host is sending or running a GIGI BASIC program.
CI		**Communications Interface:** Sets the type of communications interface to the host computer.
CI0	EIA	Use the EIA interface.
CI1	20mA	Use the 20 mA (current loop) interface.
		Set the power-up default for this set-up using the toggles on the back of the terminal.
CK		**Cursor Key:** Sets escape sequences sent by cursor (arrow) keys.
CK0	Norm	Cursor keys send normal escape sequences. These sequences move GIGI's text cursor up, down, right, and left when echoed to the terminal by the host or when the terminal is in local mode. CK0 is the default.
CK1	Appl	Cursor keys send application escape sequences. Application escape sequences are GIGI-defined escape sequences, different from the normal escape sequences, which you can use to define application-specific functions.

Using the KP and PK set-ups, you can also set escape sequences transmitted by the auxiliary keypad keys. Refer to Chapter 14, *Programming the Auxiliary Keypad*, for more information.

EM		**Expanded Character Mode:** Set text characters to normal or double width; move the text cursor to the left margin.
EM0	Off	Normal width text. EM0 is the default.
EM1	On	Double width text.
GD		**Graphic Debug:** Controls display of graphics commands on the screen for graphics programs debugging purposes.
GD0	Norm	ReGIS commands are interpreted as graphics instructions; the commands are not shown on the display. GD0 is the default.
GD1	Text	Display all ReGIS commands as text; do not execute as graphics. When GD1 is set, the string GON precedes the commands being processed and the string GOFF follows commands after ReGIS completes processing the string.
GD2	Top	Display most recently processed line of ReGIS commands as text at top of screen; execute commands.
GD3	Bot	Display most recently processed line of ReGIS commands as text at bottom of screen; execute commands. When GD2 or GD3 is set, commands appear only when there are no more commands for ReGIS to process or when the display is frozen using the NO SCROLL key. Receipt of a line feed clears the ReGIS display line.
GP		**Graphics Prefix Control:** Enables or disables a specific character as a graphics prefix character; also, allows you to change the specific graphics prefix character from the predefined exclamation mark (!) to another character. ReGIS interprets text preceded by a line feed immediately followed by the prefix character as graphics commands rather than as text until the next line feed.

GP0	Off	Disables the graphics prefix character; GIGI interprets lines preceded by the line feed/prefix character as text. GP0 is the default.
GP1	"!"	Enables the exclamation mark as a graphic prefix character. To change the prefix character from the exclamation mark, press the SHIFT key and the = (equal sign) key and then type the desired character. The graphics prefix character can be set to any of the 95 visible characters (space through tilde). Note that the graphics prefix character must be preceded by a line feed character; therefore, when the graphics prefix character is used in a file to precede a sequence of graphics commands, ensure that you precede the graphics prefix character with a line feed.
HM		**Horizontal Margins:** Adjust left and right horizontal screen margins toward the center of the screen. HM0 is the default.
HM0...9		Adjust the margin the specified number of text character spaces toward the center of the screen. This parameter selects the margins, measured in character widths, on both the right and left of the text writing area. Note that the character width is a function of the setting of the expansion mode (EM). If HM0 is selected, then there is room for 84 characters per line (42 in EM1). If HM1 is selected, 82 characters fit on a line (40 in EM1); if HM2 is selected, 80 characters fit on a line (38 in EM1), and so forth. Setting this parameter moves the text cursor to the left margin. This setting has no effect on graphics displays.
HP		**Horizontal Position:** Moves the horizontal position of the entire screen display up to 9 positions.
HP0...9		Specifies the relative position to move the screen display. When GIGI powers on, the default horizontal position is HP5. You can move the entire screen to the right by increasing HP values; you can move the screen to the left by decreasing HP values.

HS		**Hardcopy Speed:** Set the speed at which GIGI transmits the screen image to the printer or accepts data from the graphics tablet:
HS0		110 baud
HS1		300 baud
HS2		600 baud
HS3		1200 baud
HS4		2400 baud
HS5		4800 baud
HS6		9600 baud
HS7		19200 baud
IL		**Interlace:** Enables or disables scan line interlacing of the screen display. Interlacing can be used to enhance the apparent image quality on certain high persistence monitors. Interlacing also increases the quality of photographs taken of the screen. On most monitors, interlacing increases flicker and should not be used. Interlacing does not actually affect the resolution of the display.
IL0	On	Disable interlace. Display each scan line once. IL0 is the default.
IL1	Off	Enable interlace. Display each scan line twice.
KC		**Keyboard Click Control:** Enables or disables the keyclick feature of the keyboard. The keyclick feature causes most keys on the keyboard to 'click' when a key is pressed. Some keys, such as the SHIFT key, do not click.
KC0	Off	Disables keyclick; keyboard does not click at each keystroke.
KC1	On	Enables keyclick; keyboard clicks on each keystroke. KC1 is the default.
KR		**Keyboard Autorepeat:** Enable or disable the keyboard autorepeat feature for most keyboard characters. Autorepeat generates a character for as long as the key for that character is pressed. Certain keys never autorepeat.
KR0	Off	Disable keyboard autorepeat.
KR1	On	Enable keyboard autorepeat. KR1 is the default setting.

KP		**Auxiliary Keypad Functions:** Direct GIGI to set the auxiliary keypad to transmit either application escape sequences (KP1) or normal characters (KP0). Escape sequences are listed in Part 3, *Programming With GIGI*.
KP0	Norm	Send normal escape sequences for keys PF1 through PF4. For other keypad keys, send the characters printed on the keys, for example, the 5 key sends the character 5. The ENTER key sends the same code as the RETURN key. KP0 is the default.
KP1	Appl	All keypad keys (and cursor keys, if GIGI is in VT52 mode) send application escape sequences.
LE		**Local Echo:** Enables or disables local echo of keyboard input. Local echo displays each character GIGI transmits to the host system. Selection is usually based on whether communications with the host is full- or half-duplex. (Most DEC operating systems support full duplex communications and require that local echo be disabled.)
LE0	Off	Disables local echo. LE0 is the default.
LE1	On	Enables local echo; all characters transmitted to the host are also displayed on the screen or processed as control sequences.
LL		**Line/Local:** Sets GIGI either online, to a host system, or local.
LL0	Locl	Local Terminal Control; characters typed on the keyboard are displayed as typed. Characters received from the host are ignored. The local LED is lit.
LL1	OnLn	Online Host Communications; characters typed on the keyboard normally go to the host and characters received from the host are displayed. The on-line LED is lit. LL1 is the default.
MB		**Margin Bell:** Controls margin warning bell; when the cursor is moved to the position 9 characters before the right margin the terminal bell can be sounded as a warning of the approaching margin.
MB0	Off	Disables the margin bell; warning bell does not sound.

MB1	On	Enables the margin bell; bell sounds when key is struck 9 positions before the right margin. MB1 is the default.
NL		**New Line Control:** Enables or disables the generation of a line feed and carriage return when the terminal RETURN key is pressed.
NL0	Off	Carriage return <CR> moves the text cursor to the left margin on the same line. Line feed <LF> characters move the text cursor one line without change in horizontal position. NL0 is the default.
NL1	On	Carriage return generates <CR><LF>.<LF> performs a next line operation (index and move to left margin).
OS		**Overstrike:** Enable or disable overstrike feature; this feature allows multiple characters to be entered at a single screen position. This set-up is not retained by a soft reset (SHIFT/PF4) and must be manually reset after a soft reset.
OS0	On	Disable overstrike. OS0 is the normal operating mode for most operating system terminals. OS0 is the default.
OS1	Off	Enable overstrike. OS1 is used for such applications as APL and RUNOFF.
PE		**Parity Enable:** Enables or disables GIGI parity checking; selects the handling of the high-order bit of character data sent to and from the host. All characters contain 8 bits: 7 data bits and 1 parity bit, selected as either zero (and ignored on reception), even parity with even parity checking on reception, or odd parity with odd parity checking on reception. Set the power-up default for this set-up using the switches on the back panel of the terminal.
PE0	Off	No parity check; bit 8 set nonfunctional. PE0 is the default.
PE1	Even	Even parity check; bit 8 set even.
PE2	Odd	Odd parity check; bit 8 set odd.

PF		**Power Frequency:** Selects the terminal display refresh rate to match the rate supported by the monitor used. Set the power-up default for this set-up using the switches on the back panel of the terminal.
PF0	50Hz	Sets display frequency to 50 Hz.
PF1	60Hz	Sets display frequency to 60 Hz.
PK		**Programmed Keypad:** Direct GIGI to set the auxiliary keypad to transmit either standard ASCII escape sequences (PK0) or user-defined strings for a particular application (PK1). PK assignments override KP assignments.
PK0	Off	Set the keys on the auxiliary keypad to transmit escape sequences programmed for the terminal mode (VT52 or ANSI), keypad keys (KP set-up), and cursor keys (CK set-up). PK0 is the default.
PK1	On	Set the keys on the auxiliary keypad to transmit escape sequences specific to an application. Keys not programmed to send special escape sequences send normal escape sequences.
RS		**Receive Speed:** Sets the speed at which GIGI receives transmissions from the host to one of the following:
RS0		110 baud
RS1		300 baud
RS2		600 baud
RS3		1200 baud
RS4		2400 baud
RS5		4800 baud
RS6		9600 baud
RS7		19200 baud
		Set power-up values for this set-up using the switches at the rear of the keyboard.
RV		**Reverse Video:** Enable or disable reverse video.
RV0	Off	Normal video; bright characters on dark background. If disabled, the display is white (or colored) on a dark background. RV0 is the default.
RV1	On	Reverse video; dark characters on bright background. If enabled, the display is dark on a white (or colored) background.

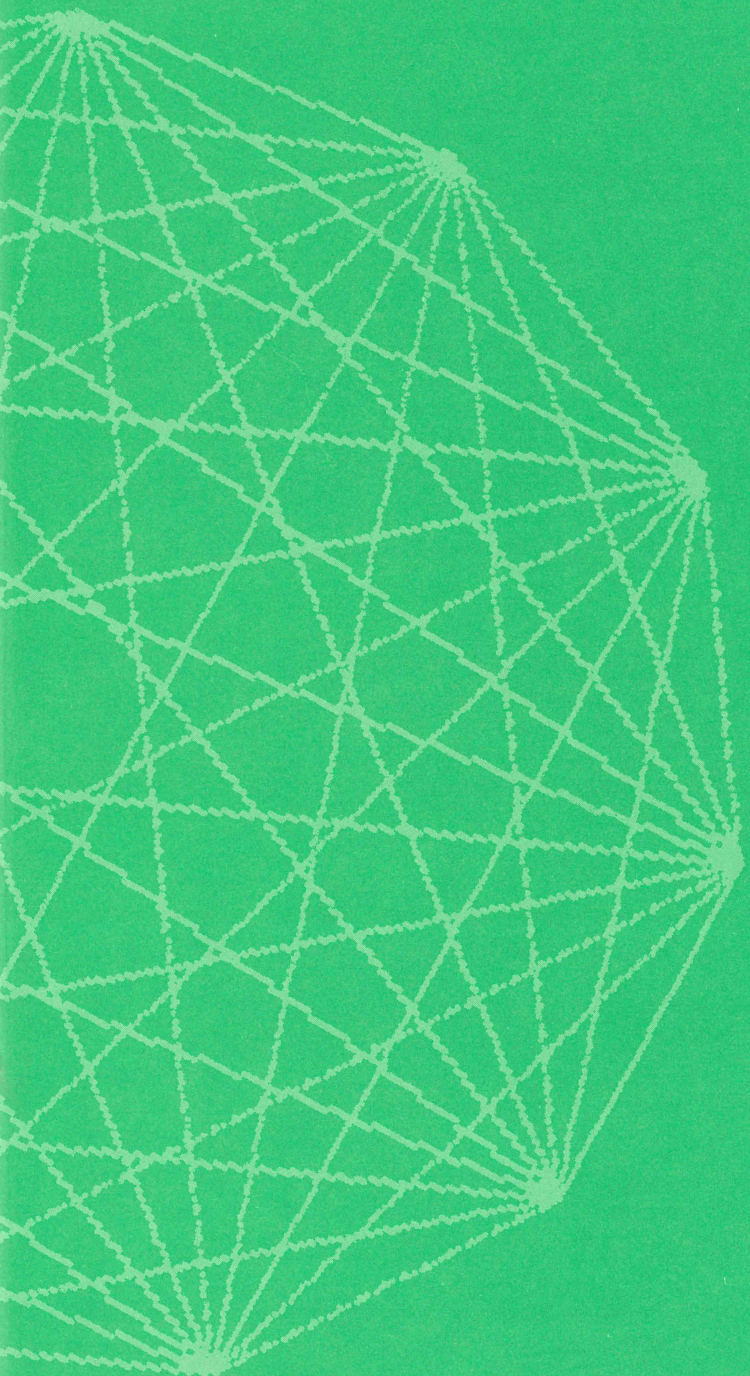
SC		**Single Character Transmission:** Enables or disables generation of a carriage return after each keystroke. Directs GIGI to transmit a carriage return automatically after each keystroke. SC is used with a host program designed to accept input in this form.
SC0	Off	Normal transmission; in most applications, the user must enter carriage return before most application programs can process user input. SC0 is the default.
SC1	On	Single character transmission; GIGI generates carriage return character after each keystroke.
SM		**Scroll Mode:** Set scrolling characteristics for the GIGI screen. This setup controls the location of the text cursor when a line feed (or another index function such as reverse index) occurs. The cursor appears at the top of the screen for a forward index and at the bottom of the screen for a reverse index.
SM0	Off	Set scrolling off; text characters transmitted after the bottom of the screen is reached are displayed on the bottom line of the screen. Cursor remains at the bottom of the display and the display is not altered.
SM1	Jump	Jump scroll Cursor moves immediately to the next line and the display is scrolled vertically to allow this line to become visible.
SM2	Smth	Smooth Scroll Cursor moves to the next line and the display is incrementally scrolled vertically to allow this line to become visible at a rate of 6 text lines-per-second. Smooth scrolling is the default. SM2 is the default.
SM3	Wrap	Wrap Scroll;when the bottom of screen is reached, text cursor moves to the top of the screen.
ST		**Terminal Self Test:** Selects and starts self tests. Values of 1 to 5 select tests to be performed. Setting the value 9 indicates that the selected tests will be performed repeatedly until failure. Setting ST0 clears all selected tests. Tests execute when the SETUP key is pressed again.
ST0	Clr	Clear selected tests.
ST1	PwUp	Select *power up* test.

ST2	HcCm	Select *hardcopy communications* test.
ST3	ExCm	Select *external communications* test.
ST4	Dspl	Select *pattern display* test.
ST5	CBar	Select *color bar* test.
ST9	Rept	Repeat selected tests to failure.
		Both the hardcopy communications test and the external communications tests require use of a loop-back connector. Refer to the document *VK100 Terminal Installation and Owner's Manual* for information on these tests.
TD		**Text Debug:** Controls the display and interpretation of special characters (line feeds, carriage returns, etc.) on the screen.
TD0	Norm	Normal display of text characters; normal processing of both text and ReGIS graphics occurs. TD0 is the default.
TD1	Text	All characters are displayed as text characters, including all control and escape sequences. No normal processing of these characters occurs, except that line feed <LF> and XON/XOFF are still interpreted for synchronization, but also are displayed as text. The representation of control codes is by the ANSI proposed standard 2-character mnemonics.
TD2	Grph	Normal display of text characters; normal control code processing occurs and those control codes which are not normally processed are displayed as graphic text. Unrecognized escape sequences are not displayed.
TL		**Tablet Locator Mode:** Enable or disable use of the hardcopy port as input to GIGI from a graphics tablet.
TL0	Off	Only the cursor keys on the keyboard can move the locator mode cross-hair. TL0 is the default.
TL1	On	When a properly-configured graphics tablet is connected to the hardcopy port, the tablet pen or cursor (as well as the cursor keys) can position the cross-hair. When TL1 is set, the tablet cursor position overrides the cursor position keys. All other keys function as usual to terminate locator mode, as will the tablet cursor button or tip switch. The hardcopy port speed (HS set-up) may require adjustment to a particular speed for your graphics tablet.

TM		**Terminal Mode:** Directs GIGI to recognize either VT52 or ANSI escape sequences.
TM0	VT52	Recognize VT52 escape sequences. In VT52 mode, only VT52 escape sequences are processed and the keyboard generates VT52 escape sequences for the cursor and numeric pad keys.
TM1	ANSI	Recognize ANSI escape sequences; only ANSI compatible escape and control sequences are processed and the keyboard generates ANSI compatible escape and control sequences.
TS		**Transmit Speed:** Sets the transmission speed of GIGI to the host to one of the following:
TS0		110 baud
TS1		300 baud
TS2		600 baud
TS3		1200 baud
TS4		2400 baud
TS5		4800 baud
TS6		9600 baud
TS7		19200 baud
		Set power-up values for this set-up using the switches at the rear of the keyboard.
UK		**United Kingdom Character Set:** Directs GIGI to use either the US character set pound sign (#) or the UK character set pound sign (&).
UK0	Off	Use the US Character set. The native character set is set initially to the USASCII character set. ReGIS character set 0 is the USASCII character set. US is the default character set.
UK1	On	Use the UK character set. G0 and G1 will refer to the UK character set, and character set 0 is loaded with the UK set. The UK set differs from the USASCII character set in that '&' appears as the UK pound sign.
		Set the power-up default for this set-up using the switches at the back of the terminal.
		To change this set-up:
		1. Alter the current set-up manually or via an escape sequence. 2. Press SHIFT/PF4 or issue <ESC>c.

VC		**Cursor Characteristics:** Selects cursor to be displayed. Only one cursor at a time is displayed. The locator mode cursor cannot be disabled.
VC0	Off	Set both text and graphics cursor off; neither cursor is visible on the screen.
VC1	Text	Set the text cursor on; cursor blinks on the screen when GIGI is in text mode. (The text cursor is a reverse video flashing block.) Cursor not visible in graphics mode.
VC2	Grph	Enables the graphics cursor; cursor is visible when GIGI is in graphics mode. (Graphics cursor is a diamond with center cross-hair.) Cursor not visible in text mode.
VC3	Both	Enables both text and graphics cursors. VC3 is the default.
VM		**Vertical Margins:** Adjust top and bottom vertical margins toward the center of the screen.
VM0...9		Adjust the margins the specified number of spaces toward the center of the screen. This parameter selects the margins, in number of character heights, both at the top and the bottom of the text writing area. If VM0 is selected, there is room for 24 lines of text; VM0 is the default. If VM1 is selected, there is room for 22 lines of text, and so forth. Setting this parameter moves the text cursor to the new top line.
XO		**Transmission Synchronization:** Specifies whether to use XON/XOFF synchronization characters.
XO0	Off	XON/XOFF not transmitted. If disabled, XON/XOFF are not transmitted, and if the terminal communications input queue should overflow, additional characters will be discarded. The SCROLL key is also disabled if XON/XOFF is disabled. This setting is not recommended for GIGI; it can be relied upon only at low receive speeds or by special host programming.
XO1	On	XON/XOFF transmitted; used for synchronization and NO SCROLL key. If enabled, XON/XOFF are transmitted to the host to control character flow to the terminals. If XON and XOFF are not honored by the host operating system, GIGI's features may be unreliable at all but the lowest receive speeds. XO1 is the default.

2



GIGI Graphics

This part of the manual describes GIGI graphics:

Chapter 3 is an introduction to GIGI graphics.

Chapter 4 describes how to execute ReGIS commands.

Chapter 5 describes how to code and use the ReGIS graphics language.

Chapter 6 describes of GIGI screen concepts.

Chapter 7 describes GIGI drawing concepts.

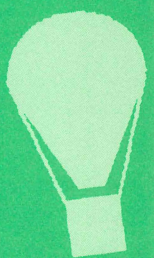
Chapter 8 describes GIGI writing concepts.

Chapter 9 describes GIGI text concepts.

Chapter 10 describes how to code and use macrographs.

Chapter 11 is a technical description of how GIGI displays images.

Chapter 12 is an alphabetically-organized description of ReGIS commands.



3 INTRODUCTION TO GIGI/ReGIS GRAPHICS

GIGI features a built-in interpreter for a graphics language called ReGIS, the Remote Graphics Instruction Set. ReGIS interprets commands that allow you to simply and efficiently control the video monitor screen and draw pictures on the screen with lines, curves, and circles. Also, ReGIS provides commands to include text characters in pictures.

THE ReGIS GRAPHICS LANGUAGE

The ReGIS graphics language is designed for conciseness and easy transport of code from the host to a ReGIS device. The language consists of commands which are modified by options. The following is a list of the ReGIS commands with a summary of the function each performs:

ReGIS Command	Functions
S	Specifies screen controls, for example, setting the screen color to blue.
W	Specifies writing controls, for example, setting the writing color to yellow.
P	Positions the graphics cursor on the screen without writing on the screen.
V	Draws vectors (straight lines) between screen locations you specify, for example, you can draw a box on the screen using just the vector command.
C	Draws curves or circles using screen locations you specify.
T	Controls display of graphics text strings and allows you to specify the characters to be displayed.
L	Controls definition and loading of alternate characters which can be displayed using the T command.
@	Define a string (for example, a ReGIS command) which can be inserted repeatedly anywhere in a ReGIS command stream; this string is both defined and invoked by the at sign (@).
R	Reports the current location of the graphics cursor; reports the name of the character set in use; reports the amount of macrograph storage in use.

In all ReGIS commands shown in the sections below, parentheses and brackets are part of ReGIS command syntax and must be coded as they are shown. The chapter *Coding and Using ReGIS Commands* contains a complete description of grammar rules for coding ReGIS commands.

DRAWING PICTURES

ReGIS provides commands for creating pictures on the video monitor screen. Pictures consist of lines, circles, curves, and text. Use the P, V, C, and T commands to draw pictures on the screen.

The example below is a BASIC program for drawing pictures on the screen. The code contains the ReGIS commands for generating various shapes on the screen, as shown in Figure 3-1. This example uses GIGI's graphics prefix character and GIGI BASIC in local mode. To enable these features, set the following set-ups:

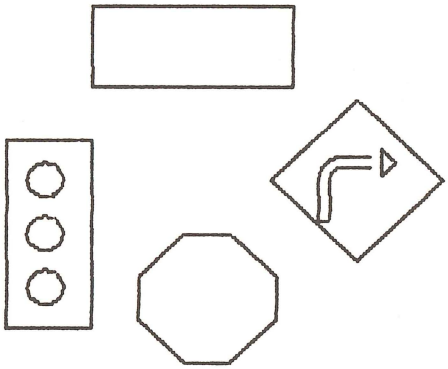
LL0	Enable Local Mode
BA1	Enable GIGI BASIC
GP1	Enable the Graphics Prefix Character

Once these set-ups are in effect, you can enter the GIGI BASIC PRINT statements to generate pictures on the screen. The PRINT statements take the form:

number PRINT " *ReGIS command string*"

where *number* is the BASIC statement number and the ! (exclamation point) is the graphics prefix character. The graphics prefix character and ReGIS command string must be enclosed in double quotes.

Figure 3-1.
Shapes You Can Draw
Using ReGIS Commands



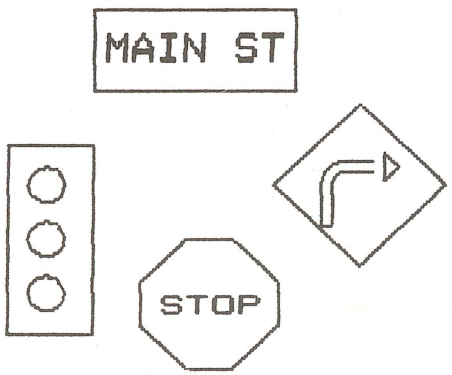
```
20   PRINT ''!s(E)''
30   PRINT ''!P[240,150] V[+140][,+60][-140][,-60]''
40   PRINT ''!P[180,250] V[][+60][,+140][-60][,-140]''
50   PRINT ''!P[365,280] V[+60,-60][+60,+60][-60,+60][-60,-60]''
60   PRINT ''!P[442,257] V[,+20][+10,-10][-10,-10]''
70   PRINT ''!P[305,320] V[+32][+32,+32][,+32][-32,+32][-32]''
80   PRINT ''![-32,-32][,-32][+32,-32]''
90   PRINT ''!P[207,280] C[+13]''
100  PRINT ''!P[207,320] C[+13]''
200  PRINT ''!P[207,360] C[+13]''
300  PRINT ''!P[406,310] V[,-30]C(CA-90)+[+10]V[+18]''
400  PRINT ''!P[406,310] V[-8][,-30]C(CA-90)+[+18]V[+18]''
```

Including Text Strings in Pictures

A graphic text string is a string of characters from any of GIGI's four character sets displayed on the monitor when GIGI is in graphics mode. A character is an eight-by-ten dot pattern which is the basic unit of each of GIGI's four character sets. Figure 3-2 shows the example shapes labeled with text strings; the T commands used to generate the text string are highlighted.

In this example (also in GIGI BASIC) the CHR$ function is used to include the escape character in the PRINT statements. Note that the Pp (upper- and lower-case P) now precede the ReGIS command in the quoted string.

Figure 3-2.
Examples of GIGI Text Strings in Pictures




```
200  PRINT CHR$(155)+''Pp''
210  PRINT ''s(E)''
300  PRINT ''P[240,150] V[+140][,+60][-140][,-60]''
400  PRINT ''P[180,250] V[][+60][,+140][-60][,-140]''
500  PRINT ''P[365,280] V[+60,-60][+60,+60][-60,+60][-60,-60]''
600  PRINT ''P[442,257] V[,+20][+10,-10][-10,-10]''
700  PRINT ''P[305,320] V[+32][+32,+32][,+32][-32,+32][-32]''
800  PRINT ''V[-32,-32][,-32][+32,-32]''
900  PRINT ''P[207,280] C[+13]''
1000 PRINT ''P[207,320] C[+13]''
2000 PRINT ''P[207,360] C[+13]''
3000 PRINT ''P[406,310] V[,-30]C(CA-90)+[+10]V[+18]''
4000 PRINT ''P[406,310] V[-8][,-30]C(CA-90)+[+18]V[+18]
4100 PRINT ''P[246,165] T(s2)(h3) 'MAIN ST' ''
4200 PRINT ''P[285,359] T(s2)(h2) 'STOP' ''
5000 PRINT CHR$(155)+''!''
```

Statements 4100 and 4200 contain the T commands that generate the graphic text strings ''MAIN ST'' and ''STOP''.

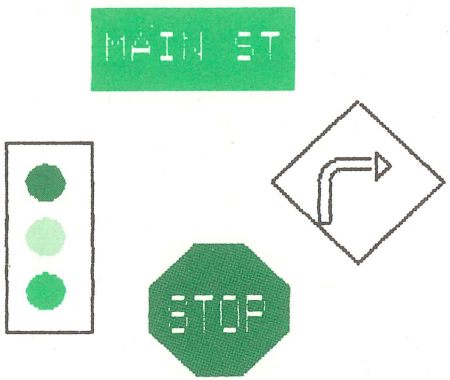
Controlling Screen and Writing Functions

The S command controls functions related to the entire screen display; the W command controls functions GIGI uses in writing on the screen. In the example program, statement 10 contains an S command that erases the entire screen. Using the S command, you can also set the color (or gray level, if you have a black-and-white monitor) of the screen.

The W (writing) command lets you set the writing color to contrast with the screen color. The W command also provides functions that allow you to shade an object on the screen. Figure 3-3 shows the street sign shaded and colored yellow. Statement 50 contains the ReGIS W command to turn on shading and write in yellow; statement 70 turns shading off and returns the writing color to white. This program contains other W commands which enable and disable writing modes; these functions are described in Chapter 8, "Writing Concepts."

This example uses the GIGI BASIC GON$ (graphics on) function to enable graphics mode. The GOFF$ function at the end of the program disables graphics mode and returns GIGI to text mode.

Figure 3-3. Examples of GIGI's Screen and Writing Commands




```
10    PRINT GON$,"s(e)"
40    PRINT "P[240,150] "
50    PRINT "W(S1 I(Y))"
60    PRINT "V[+140][,+60][-140][,-60]"
70    PRINT "W(S0 I(W))"
80    PRINT "P[246,165] T(s2)(h3)(w(c)) 'MAIN ST' "
90    PRINT "P[180,250]"
100   PRINT "V[][+60][,+140][-60][,-140]"
110   PRINT "P[207,280] W( I(R) S1) C[+13] W(S0)"
120   PRINT "P[207,320] W( I(Y) S1) C[+13] W(S0)"
```

continued next page

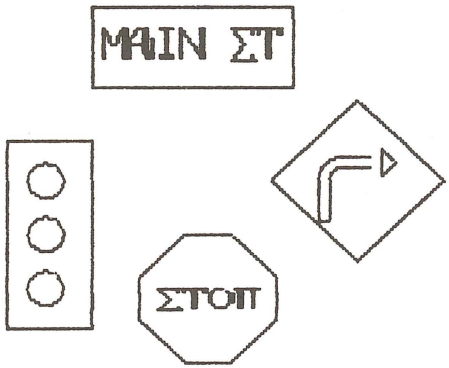
```
130  PRINT ''P[207,360] W( I(G) S1) C[+13] W(S0)''
140  PRINT ''p[280,390]''
150  PRINT ''w(s1)''
160  PRINT ''v[+32,+32][+32,+0][+32,-32][+0,-32]''
165  PRINT ''[-32,-32][-32,+0][-32,+32][+0,+32]''
170  PRINT ''P[290,359] T(s2)(W(C))(h2) 'STOP' ''
180  PRINT ''W(S0 I(Y))''
190  PRINT ''P[365,280]''
200  PRINT ''V[+60,-60][+60,+60][-60,+60][-60,-60]''
210  PRINT ''P[438,258]''
220  PRINT ''V[,+20][+10,-10][-10,-10]''
230  PRINT ''P[406,310] V[,-30]C(CA-90)+[+10]V[+18]''
240  PRINT ''P[406,310] V[-8][,-30]C(CA-90)+[+18]V[+18]''
250  PRINT ''W( I(W) R)''
260  PRINT GOFF$
```

CHARACTER SETS

GIGI supports the full 127-character ASCII character set and can display the 95 visible characters. For graphic displays, GIGI uses the visible 95-character set, each unit of which is a two-dimensional dot pattern GIGI can display as a character. GIGI provides four character sets, a native ASCII character set and three alternate character sets. GIGI's native character set is the 95-character ASCII character set from space through tilde, as listed in Table 1-1. You can set the native character set to be either US ASCII or UK ASCII.

The three alternate character sets are not set initially and appear as filled-in blocks. Using the ReGIS L (load) command, you can set the dot patterns that make up the characters to create characters for a given application. The sample program below shows the BASIC code required to load Greek characters into the uppercase characters MAIN ST and STOP into alternate character set 2. Figure 3-4 shows the street signs labeled with these Greek characters.

Figure 3-4.
Examples of Characters From an Alternate Character Set




```
10    PRINT CHR$(155)+''Pp''
15    PRINT ''s(E)''
20    PRINT '' L(a2)' '0000000000000000;''
10    PRINT '' L(a2)'A'1E264686FE868783;''
20    PRINT '' L(a2)'E'FF8180F0808081FF;''
30    PRINT '' L(a2)'I'3E0808080808083E;''
40    PRINT '' L(a2)'M'81C3A5A599818181;''
50    PRINT '' L(a2)'N'81C1A19189858381;''
60    PRINT '' L(a2)'R'FC828181FE808080;''
70    PRINT '' L(a2)'S'FF41201008102041FF;''
80    PRINT '' L(a2)'T'FF99181818181818;''
300   PRINT ''P[240,150] V[+140][,+60][-140][,-60]''
400   PRINT ''P[180,250] V[][+60][,+140][-60][,-140]''
500   PRINT ''P[365,280] V[+60,-60][+60,+60][-60,+60][-60,-60]''
600   PRINT ''P[442,257] V[,+20][+10,-10][-10,-10]''
700   PRINT ''P[305,320] V[+32][+32,+32][,+32][-32,+32][-32]''
800   PRINT ''V[-32,-32][,-32][+32,-32]''
900   PRINT ''P[207,280] C[+13]''
1000  PRINT ''P[207,320] C[+13]''
2000  PRINT ''P[207,360] C[+13]''
3000  PRINT ''P[406,310] V[,-30]C(CA-90)+[+10]V[+18]''
4000  PRINT ''P[406,310]V[-8][,-30]C(CA-90)+[+18]V[+18]''
4100  PRINT ''P[246,165] T(s2)(a2)(h3) 'MAIN ST' ''
4200  PRINT ''P[285,359] T(s2)(a2)(h2) 'STOP' ''
5000  PRINT CHR$(155)+'\''
```

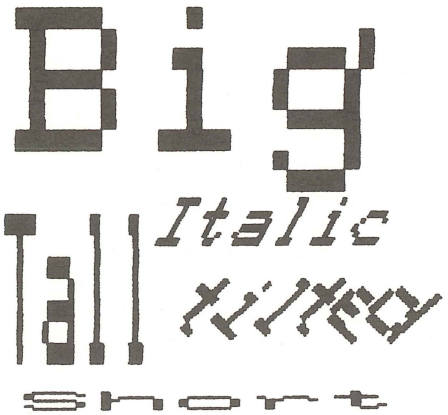
In this program, lines 20 through 80 load Greek characters into alternate character set 2 (a2); lines 4100 and 4200 direct GIGI to display text using the alternate characters.

GRAPHICS ATTRIBUTES

An attribute is a characteristic or distinctive feature of an object; for example, color or size. ReGIS provides the means for assigning attributes to the screen, to writing, and to text.

The examples above showed how to set attributes for the screen and writing by means of the S and W commands. You can also set attributes for text. GIGI provides various ways to set attributes for text; the GIGI BASIC program below demonstrates some of the attributes you can set for text using the T command.

Figure 3-5.
Example of Graphic Text Attributes




```
10    PRINT CHR$(155)+''Pp''
20    PRINT ''w(m20)''
30    PRINT ''s(e)p[165,135]''
40    PRINT ''t(a0 s10)'Big'66444444''
50    PRINT ''t(s3 h16)'Tall'0''
60    PRINT ''t(s3 i-27)'Italic'6644444444444''
70    PRINT ''t(s3 i0 d-27)'tilted'6666666644444444''
80    PRINT ''t(e s6 d0 h2)'Short' ''1000 PRINT CHR$(155)+''\''
```

Figure 3-5 illustrates text attributes of size, height, slant, and direction. Also, this program shows ReGIS ability to move the current location in a numbered direction. In the example, the numbers following the text commands direct GIGI to move the graphics cursor in direction 6 (down) or direction 4 (backward) before performing the next graphics operation. Chapter 7, *Drawing With ReGIS*, includes more examples of how to use these numbered directions, which are referred to as offset directions.

Attributes are set by means of command options (discussed in Chapter 5). GIGI sets default attributes when powered up; these attributes become the current attributes for screen, text, or drawings until you reset them with appropriate command options.

4 EXECUTING ReGIS COMMANDS

To execute ReGIS commands, GIGI must be in graphics mode. Graphics mode is that GIGI operating mode in which the ReGIS interpreter is interpreting all characters transmitted from the local keyboard, from GIGI BASIC, or from the host operating system.

This chapter describes how to execute ReGIS commands in local mode, using GIGI BASIC statements, and using host operating system resources.

ENTERING AND EXITING GRAPHICS MODE

You can use either of two methods for entering graphics mode from text mode: a device control string (DCS) or the GIGI graphics prefix character. (Device Control Strings are described in detail in the Chapter *Programming GIGI Using Escape Sequences*.) The following sections describe both methods.

Device Control Strings

Device Control Strings (DCS) are special character sequences that direct a device to perform a specific function. GIGI uses device control strings to enter and exit graphics mode. The device control string which directs GIGI to enter graphics mode takes the form:

<ESC>Pp *ReGIS commands...*

The <ESC> in this sequence is the escape character; <ESC>P (uppercase letter P) is the DCS prefix and the lowercase letter "p" specifies that this is a graphics DCS. All the characters following the DCS prefix are interpreted as ReGIS commands.

The device control string which directs GIGI to exit graphics mode takes the form:

<ESC>\

The <ESC>\ (escape and backslash characters) comprise the DCS terminator. GIGI returns to text mode when it receives this terminator.

Line Feed and Graphics Prefix

The graphics prefix character is a special character which, like the device control string, directs GIGI to enter graphics mode. The graphics prefix character is set by means of the GP (Graphics Prefix) SET-UP parameter.

The default prefix character is the exclamation point (!), but, in set-up mode you can reset the character by pressing the SHIFT key and the equal key, then entering the character you desire. When GP1 is set, the GIGI interprets any characters following the graphics prefix character as ReGIS commands.

The next line feed character returns GIGI to text mode.

To execute a ReGIS command, the graphics prefix character must be preceded by a line feed character.

EXECUTING ReGIS COMMANDS

GIGI can execute ReGIS commands entered from a variety of input sources. The following sections describe these methods for entering ReGIS commands:

- Directly from the keyboard
- Using GIGI BASIC statements
- Using host operating system resources such as programming languages, text editors, graphics editors, and applications packages

Executing ReGIS Commands Entered Directly from the Keyboard

To execute ReGIS commands transmitted directly from the keyboard, place GIGI in local mode (set-up LL0). Then type Device Control String prefix:

<ESC>Pp

This places GIGI in graphics mode and allows you to enter ReGIS commands that GIGI interprets and executes immediately.

To display the commands as GIGI executes them, use the graphics display set-up (GD2 or GD3), which directs GIGI to display data entered at the terminal on the top or bottom of the video monitor screen.

Executing ReGIS Commands Using GIGI BASIC Statements

You can use GIGI BASIC statements to execute ReGIS command strings. Enable GIGI BASIC (set-ups BA1 and LL0) and the graphics prefix (set-up GP1), then enter the GIGI BASIC statements in the form described in Chapter 3, for example:

```
10      PRINT "!P[200,200]V[+200,][,+200][-200,][,-200]"
```

The ReGIS commands in this statement draw a square beginning at screen location [200,200].

As described in Chapter 3, you can also use GIGI BASIC intrinsic functions to enable graphics mode. The GON$ function enables graphics mode; the GOFF$ function disable graphics mode.

Executing ReGIS Commands Using Host Operating System

Just as you can direct ReGIS commands to GIGI from the keyboard in local mode and using GIGI BASIC statements, you can also use host operating system facilities. For example, such operating system facilities as a command language or programming language (for example, FORTRAN or PASCAL) can transmit graphics commands to be interpreted by ReGIS. GIGI intercepts the <ESC>Pp from the host just as it intercepts the <ESC>Pp directly from the keyboard.

In many cases, you can use operating system commands to transmit graphics commands to GIGI. For example, the following DCS, including ReGIS commands, reside in a file called HOST.PIC in host on-line storage:

```
<ESC>Pp P[200,200] V[+200,][,+200][-200,][,-200] <ESC> \
```

This sequence directs GIGI to draw a square beginning at location [200,200]. If you are using the VAX/VMS operating system, you can transmit the file to GIGI using the VMS TYPE command:

```
TYPE HOST.PIC
```

5 CODING AND USING ReGIS COMMANDS

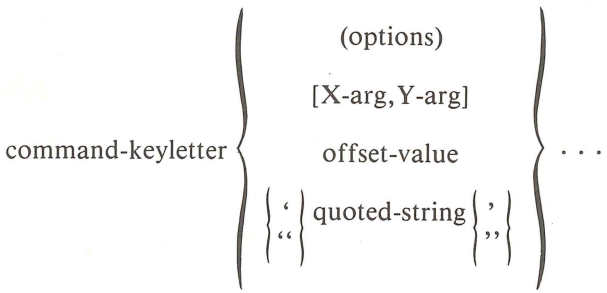
This section describes the elements that comprise a ReGIS command, grammar rules for coding the commands, and a description of how ReGIS interprets commands.

ReGIS COMMAND ELEMENTS

ReGIS commands are composed of the following elements:

- A command keyletter which specifies the operation you want ReGIS to perform.
- Parenthesized command options which set attributes to be used in subsequent graphics processing.
- Bracketed pairs used to specify two-dimensional information.
- Numeric values that specify cursor movement relative to the current cursor position.
- Punctuation symbols such as commas and semicolons.
- Quoted strings used to specify characters to display as graphics text.

The general form of a ReGIS command is:



Except for the command keyletter, each of the elements in the above is optional. The sections below describe each of these command line elements.

Likewise, except for the command keyletter, you can code these syntax elements in any order you wish. GIGI interprets the elements in the order in which you code them; this order is likely to affect the net effect of the graphics display.

Command Keyletters

A command keyletter specifies the operation you want ReGIS to perform. Commands take the form of a single letter which is a mnemonic for the graphics operation to be performed:

Command Keyletter	Command Name	Function
S	Screen	Specify screen initialization functions.
W	Writing	Specify writing (that is, line and text drawing) initialization functions.
P	Position	Position the graphics or text cursor to specific pixel location.
V	Vector	Set the end point for a line.
C	Curve	Define a point in a curve; specify the radius of a circle.
T	Text	Specify text to be displayed.
L	Load	Load a character set.
R	Report	Report the status of macrographs, current pixel location, or enter locator mode.
@	Macrograph	Define, invoke, or delimit a macrograph.

Command Options

Options are parenthesized command elements that direct GIGI to assign an attribute for a given display operation, that is, a drawing or text display. Attributes are features of or qualities associated with a graphics object at the time that the object is written to the screen. For example, a line on the screen may be blinking and may be colored green. Both *blink* and *green* are attributes of the line.

You can enter options one at a time within parentheses or as a parenthesized list. For example, ReGIS parses both of the following options specifications identically:

```
T(S3 I45 D-27 H40)
```

or

```
T(S3)
  (I45)(D-27)(H40)
```

Bracketed Pairs

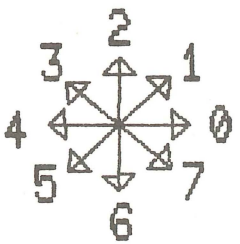
Bracketed pairs are specifiers which use two numbers to specify screen locations and other two-dimensional data; they are always specified in the form:

[X-arg,Y-arg]

The value to the left of the comma always references a horizontal value (from left to right); the value to the right of the comma always references a vertical value (from top to bottom). Using these bracketed pairs, you can reference screen coordinates as points on the screen (that is, screen locations) or as [width,height] text size.

Offset Directions

An offset direction is an unpunctuated (that is, takes no parentheses or no brackets) value which specifies movement in a direction relative to the current cursor location. Valid number entries are 0 through 7, which correspond to the following directions:



Offset directions are used with the P, V, C, and T ReGIS commands to move the graphics cursor before display of graphics objects, for example a text string.

Quoted Strings

The T (Text) command operates on quoted strings which contain characters GIGI displays when the T command executes. ReGIS parses both single and double quotes identically.

In cases where you want to include a single quote in a string, you can use the double quotes to enclose the string (and vice versa). Also in this case, you can double the enclosed quote character. ReGIS uses the first quote character after the T command keyletter as the string delimiter.

Punctuation Rules for ReGIS commands

ReGIS commands are free-form; blanks are not delimiters and GIGI ignores them except in graphics text strings. Likewise, GIGI ignores commas except in coordinate specifications. You can spread a command across as many lines as seem appropriate.

The following syntax conventions apply for coding ReGIS commands:

( )	**Parentheses:** Parentheses enclose options (attributes) lists. The list of attributes is associated with the current (that is, most-recently interpreted and executed) command keyletter.
[ ]	**Brackets:** Brackets are used to enclose pairs of values that reference two-dimensional entities such as a screen coordinate location ([column,row]) or the size of a text character ([width,height]). Bracketed pairs are used both at the command level and within attributes lists.
" "	**Quotes:** Quotes (either single or double quotes) are used to enclose text strings to be displayed by ReGIS.
,	**Commas:** Commas separate the two-dimensional values in bracketed pairs and, optionally, can be used to separate multiple attributes within an attributes list.
;	**Semicolon:** Semicolons have two uses: first in the definition of macrographs, second in terminating execution of a command.
;" "	**Comments:** There are no comments as such. However, you can use a semicolon and quotes to enclose a comment string; generally, ReGIS ignores this construction.

Attribute Currency and Scope

ReGIS interprets and executes commands in the sequence in which you enter them. Therefore, the most recently-executed command of a given type becomes *current*, that is, applies until another command of the same type is executed. The S (screen), W (writing), and T (text) commands set attributes as they are executed.

GIGI maintains attributes set by these commands as the current attributes. For example, if you set the screen background color to blue, it remains blue until you enter another S command to reset the screen background attribute. The scope of the screen command applies for all commands which follow it.

The W command sets attributes which can be overridden in subsequent Vector, Curve, and Text commands by temporary writing controls. These default conditions cannot be permanently reset except by another Write command. However, the temporary conditions apply only for the scope of the Vector, Curve, or Text command in which they are encountered.

Use the following settings for the S, W, and T commands to set attributes to their initial power-up values:

Screen	Writing	Text
E	I7	A0
I0	P1	I0
	(M2)	D0
	A0	S1
	N0	
	S0	

Error Handling

ReGIS does not display error messages. If you make an error in specifying a command, you must figure out your error by yourself based on the results of the command execution. If the command didn't execute the way you wanted it to, you probably dropped a closing parenthesis (a common mistake) or embedded a command in an inappropriate place in your code.

Remember, ReGIS interprets just what you send to it!

6 SCREEN CONCEPTS

The video monitor connected to GIGI displays the contents of GIGI's video memory, which maintains a collection of pixels, or picture elements. A picture element is the smallest resolvable, that is, physically distinguishable, part of the video memory.

GIGI draws pictures on the screen by setting (turning on) pixels. That is, when several pixels in a row are set, a line appears on the screen. To let you directly specify which pixels to set, GIGI uses an [X,Y] coordinate system.

A coordinate is a number used to specify a location on a line. X-coordinates specify horizontal locations; Y-coordinates specify vertical locations. This coordinate system allows you to specify discrete points on the screen. A point is the intersection of a horizontal line and a vertical line. Figure 6-1 shows a typical coordinate system with [X,Y] locations specifying a point in the system.

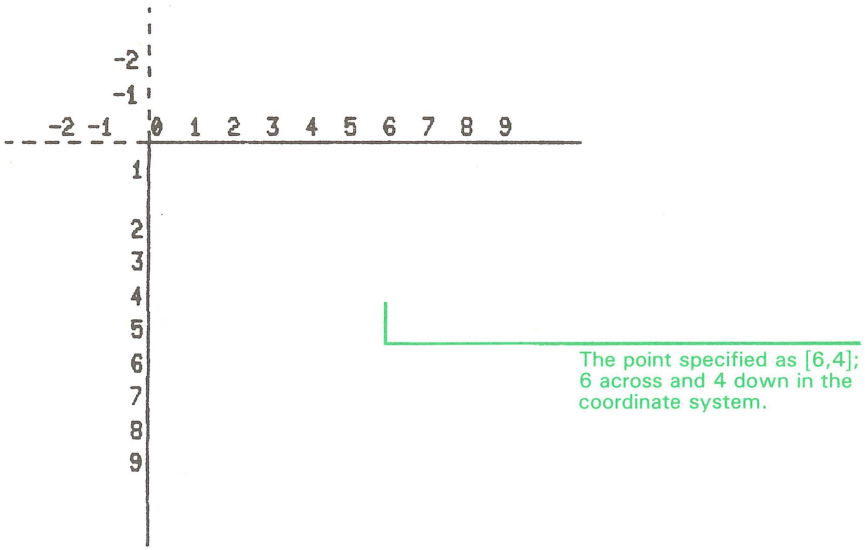


Figure 6-1.
A Typical Coordinate System

GIGI'S COORDINATE SYSTEM AND VIDEO MEMORY

GIGI uses a bit map in video memory to map points in the coordinate system and the pixels that comprise the screen. For GIGI, a bit map is an area of microprocessor memory in which each bit corresponds to a pixel on the monitor screen. Figure 6-2 shows the screen, the coordinate system, and an image (a rectangle) on the screen.

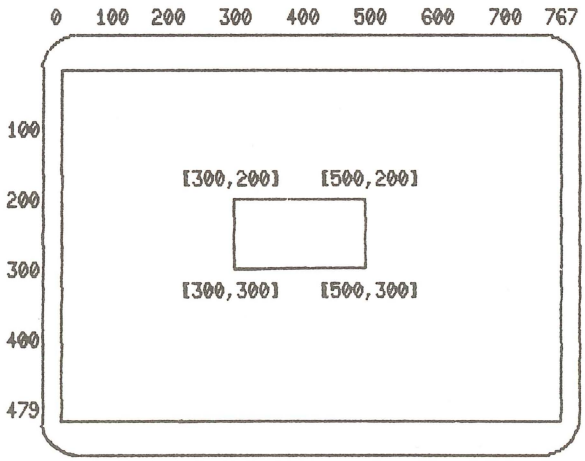


Figure 6-2.
Screen Layout, Coordinate System, and a Screen Image

Referring to Points on the Screen

The coordinate system allows you to refer to each point on the screen by means of its X- and Y-coordinates. ReGIS requires brackets around the screen location:

[X,Y]

You can specify values for X and Y explicitly by entering a number; if you do not enter a number, GIGI uses the current value for X or Y.

GIGI begins counting screen locations at the top left corner of the screen. This location is called the screen origin; it is location [0,0].

X-coordinates range from 0 (the left edge of the screen) to 767 (the right edge of the screen). Y-coordinates range from 0 (the top of the screen) to 479 (the bottom edge of the screen).

This set of locations is the working coordinate set; GIGI allows you to refer to points up to Y-coordinate 512, which is below the bottom of the monitor screen.

Points and Pixels

As mentioned above, a pixel, or picture element, is the smallest physically distinguishable part of the monitor screen. GIGI maps two vertical points, each odd-even pair, to a pixel. For example, points [200,340] and [200,341] correspond to the same pixel, the even pixel at location [200,340]. GIGI lets you reference 768 pixels on each horizontal line and 240 pixels on each vertical line. Figure 6-3 illustrates how GIGI maps two points to each pixel.

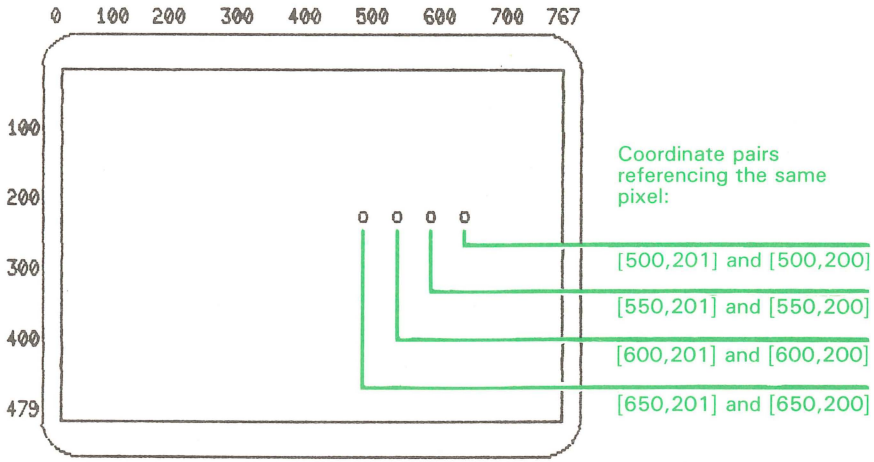


Figure 6-3.
Mapping Points to Pixels on the Monitor Screen

CURRENT LOCATION AND THE GRAPHICS CURSOR

The current location is the point on the screen most recently moved to or drawn to. GIGI displays a graphics cursor which shows you the screen location GIGI is referencing. The graphics cursor is a diamond-shaped display with center cross-hairs:

The graphics cursor is displayed only when GIGI is waiting for the next character of input.

The graphics cursor moves to let you see changes to the current location. For example, when you enter a ReGIS command to move from location [150,150] to location [200,150], the graphics cursor moves, along with the current screen location, to [200,150].

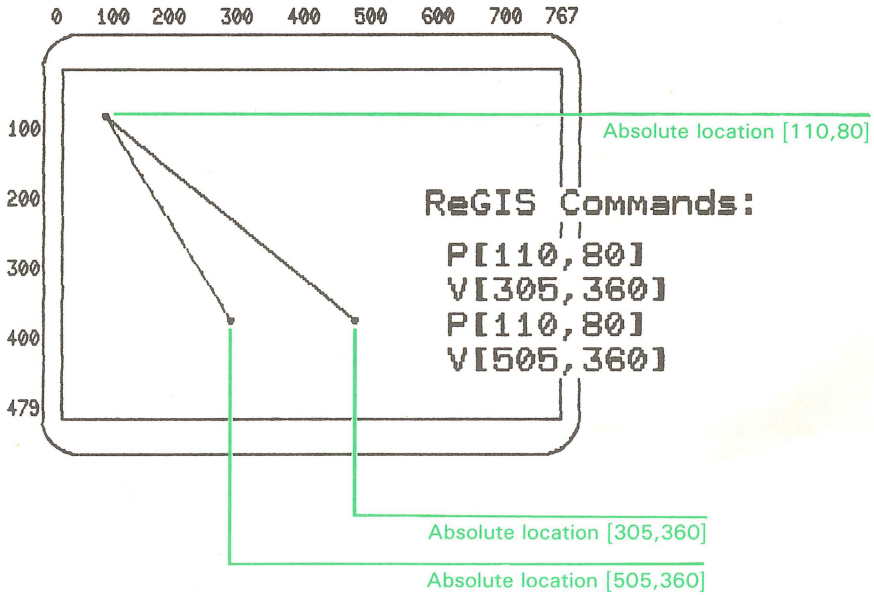
You can direct GIGI to move the cursor and the current location by specifying locations in ReGIS commands. ReGIS allows you to reference absolute locations and relative locations.

Absolute Locations

Absolute locations specify points in the coordinate system which are based on the screen origin location, [0,0]; they are specified as unsigned [X,Y] values (that is, [150,150] rather than [+ 150,-150]). For example, when you specify the absolute location [150,150], you are referencing the specific point 150 across and 150 down from screen location [0,0].

Use absolute locations to draw lines from specific screen locations to other specific screen locations, for example from one corner of the screen to the other.

Figure 6-4.
Using Absolute Screen Locations to Draw Lines



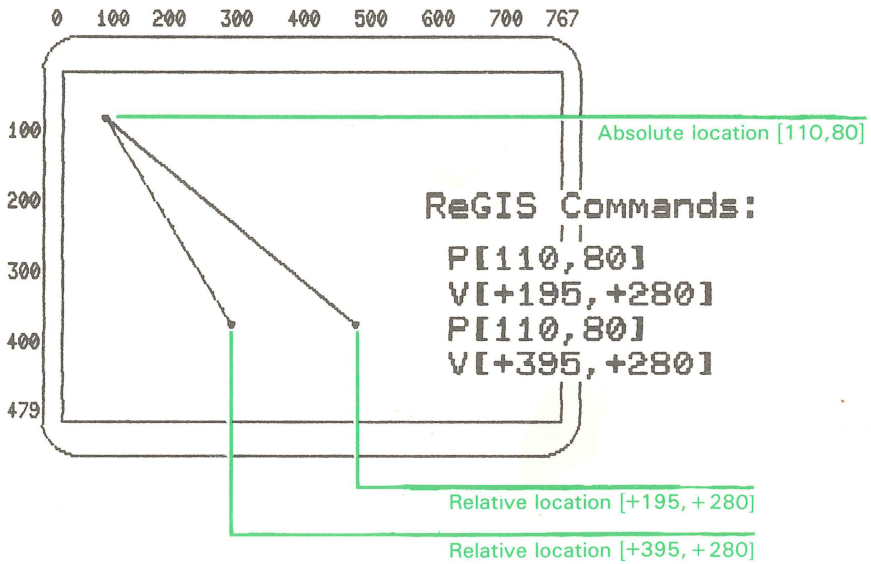
Relative Locations

Relative locations specify a distance in pixels from the current location. For example, you can use the current location and a relative location to draw a line 100 pixels long beginning at the current location. The direction in which GIGI draws the line depends on the sign of the [X,Y] values:

+X Move the current location to the right.
-X Move the current location to left.
+Y Move the current location down.
-Y Move the current location up.

You can enter combinations of signed [X,Y] values (that is, relative locations) to direct the cursor anywhere on the screen.

Figure 6-5.
Using Relative Screen Locations to Draw Lines



Relative locations are useful in situations where the origin location for a drawing is not as significant as the proportions of the drawing.

Combining Absolute and Relative Locations

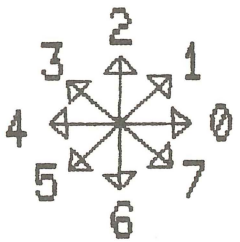
You can specify a relative (signed) coordinate with an absolute (unsigned) coordinate in the same bracketed pair. For example, [100,+100] moves the current location to absolute X-coordinate 100 and to the Y-coordinate 100 points down from the current Y-coordinate.

POSITIONING THE CURSOR ON THE SCREEN

You can control the graphics cursor and the current location using the ReGIS P (position) command. For example, by entering the following P commands, you could position the cursor to each of the locations shown in Figure 6-2, above.

p[300,200]
p[300,300]
p[500,300]
p[500,200]

Also, you can position the cursor you can position the cursor using one of the following numbers, which correspond to directions offset from the current location:



For example, the command:

P6644222

moves the current location without writing to the screen. This command moves the current location down (66), left (44), and up (2222).

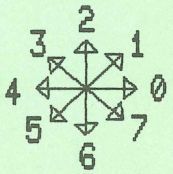
GIGI moves the current location 1 pixel for each value entered. The writing command M option allows you to specify a multiplier for the direction value. For example:

P(W(M8))6644222

moves the current location 8 times for each value specified.

Table 6-2 summarizes the functions of the position command.

Table 6-1. Position Command Functions

Function	Position Command	Comments
Move current location to location specified by [*X-loc, Y-loc*].	P[*X-loc, Y-loc*]	
Move location in specified direction.	P *direction*	*direction* can be one of the following: 
Set temporary writing controls.	P(W(*writing options*))	See description of W (writing) command options.
Save current location on the stack.	P(B)	
Restore most recently-entered location from stack.	P(E)	

SCREEN COLOR ATTRIBUTES

GIGI sets the intensity (color) attribute to the screen. If you have a color monitor, you can assign a background color to the screen using one of the following eight colors:

Dark	Green
Blue	Cyan
Red	Yellow
Magenta	White

For black and white monitors, these colors translate to degrees of brightness referred to as the gray scale.

You can also assign colors to be used in drawing and text display; these concepts are discussed in the chapters *GIGI Writing Concepts* and *GIGI Text Concepts*, below.

ERASING THE SCREEN

In general when you are creating an image on the screen, you want a blank screen to draw on. Use the S command E (erase) option to erase any writing on the screen and to set the current location on the screen to [0,0].

SCREEN ADDRESSING

The screen you can see on the video monitor is a subset of a larger logical coordinate system. This larger coordinate system is, conceptually, a two-dimensional grid, each location of which is a pixel which can be referenced by its [X,Y] location. The upper-left limit of the coordinate system is approximately [-32000,-32000]; the lower-right limit of the coordinate system is approximately [+ 32000, + 32000].

When you specify addresses that pass these limits, GIGI wraps back to the original locations and overlays any images that exist in the original locations.

By default, GIGI uses a subset of this system to address a screen 767 locations across and 479 locations deep. The default top-left pixel of the screen is addressed at location [0,0]; the default bottom-right pixel is addressed at location [767,479]. Figure 6-6 shows how the screen GIGI uses by default relates to the coordinate system.

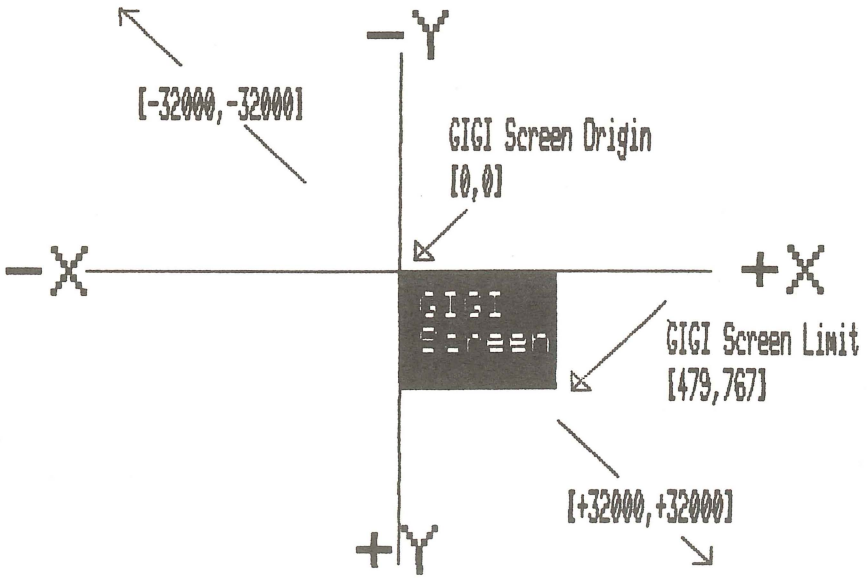


Figure 6-6. Relationship of GIGI Screen to Coordinate System

Using the Screen Addressing Options

The A (addressing) option is used to allow the screen to be addressed as if it were of a different size or orientation than it actually is.

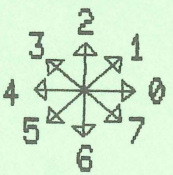
There is no restriction on the relative sizes of the left, right, top and bottom margins. If the right margin value is less than the left margin value, then the X-coordinate increases to the left and similarly for top and bottom margins. You can reset the values of the screen coordinates if necessary; this is, however, not recommended. For example, the following command defines the scope of the logical addresses using [0,0] as the origin and [100,100] as the lower-right limit of the screen. That is, after defining this addressing, a line drawn from [0,0] to [100,100] goes from the top left corner of the screen to the bottom right corner of the screen:

```
S(A[0,0][100,100])
```

THE S (SCREEN) COMMAND

The ReGIS S (Screen) command provides controls for many of GIGI's screen-handling functions. Table 6-2 shows examples of these functions and lists the format of the screen command for coding the function. The chapter *ReGIS Command Descriptions* provides a detailed description of the S command.

Table 6-2. Summary of Screen Command Functions

Function	Screen Command	Comments
Erase the screen.	S(E)	
Color the screen.	S(I(D)) or S(I0) S(I(B)) or S(I1) S(I(R)) or S(I2) S(I(M)) or S(I3) S(I(G)) or S(I4) S(I(C)) or S(I5) S(I(Y)) or S(I6) S(I(W)) or S(I7)	Dark Blue Red Magenta Green Cyan Yellow White
Set negative screen	S(N1)	Reverse Video
Set normal screen	S(N0)	Normal Video
Set time before next ReGIS command execution.	S(T *ticks*)	*ticks* = 50th or 60ths of second.
Adjust screen coordinate addressing limits.	S(A[X-loc,Y-loc] [X-loc,Y-loc])	
Specify vertical portion of screen to print.	S(H[,0][,479])	Prints entire screen.
Move contents of screen to a specific point.	S[0,0][767,479]	
Move contents of screen in a specific direction.	S *direction*	*direction* can be one of the following: 

7 DRAWING WITH ReGIS

GIGI provides commands that allow you to create pictures by drawing on the screen display. Pictures consist of lines, curves, circles, and text.You draw using the P (position), V (vector), and C (curve) commands; you display text on the same screen using the T (text) command. This chapter describes the commands and basic techniques for drawing lines, curves, and circles; Chapter 9, *Text Concepts*, describes how to include text in drawings. The W (writing) command, described in the chapter Writing Concepts provides options that modify and enhance lines, curves, and text.

DRAWING LINES

You draw lines by entering the V command with screen locations specifying where a line begins and ends.Table 7-1 lists the functions of the V command.

Drawing Lines Using Locations

In many cases, you draw lines by positioning the graphics cursor to the point at which you want the line to begin,then draw from that point to an end point. For example:

P[100,100]

V[200,200]

draws a line from location [100,100] to location [200,200]. Each location specified in the V command string draws a line from the current location. Figure 7-1 shows a ReGIS V command string consisting of several points. ReGIS begins drawing at location [300,100] and completes the vector at location [300,388].

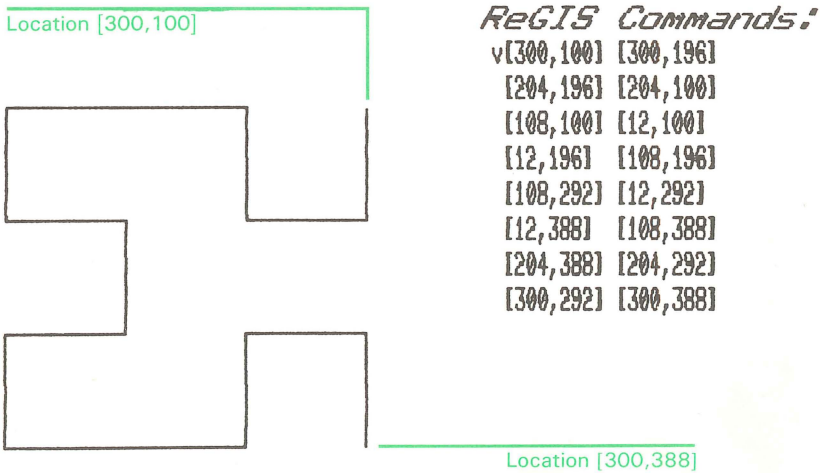
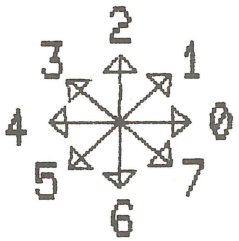


Figure 7-1. Example of a V Command Using Screen Locations

Drawing With Offset Directions

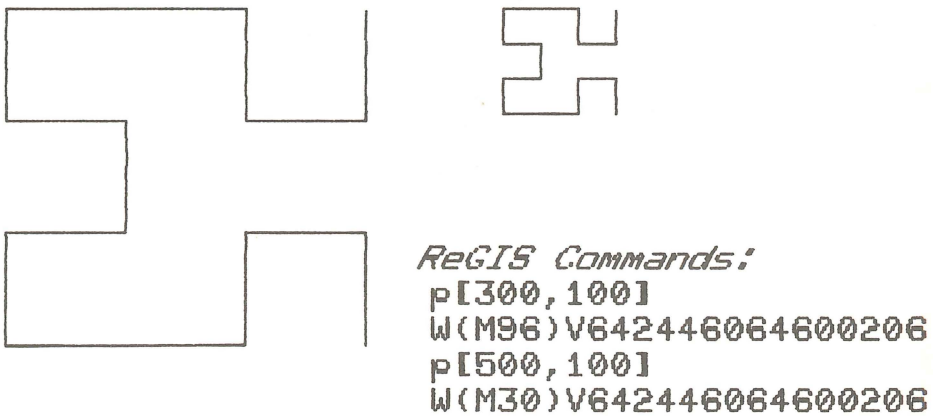
As mentioned above, you draw straight lines using locations specified in a the V (vector) command. You can also draw lines using the following directions, which are offsets from the current location:



You can draw lines (with the V command) or move the current location (with the P command) using one of these directions rather than a screen location.

Also, using the W (write) command multiplier option, (W(M)), you can direct GIGI to replicate each pixel drawn in a V command or moved with the P command. Figure 7-2 shows how to use the offset direction and the writing multiplier to draw vectors.

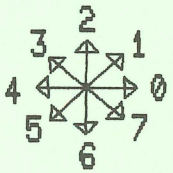
Figure 7-2.
Drawing Vectors With Offset Directions and Writing Multipliers



The first drawing replicates each pixel drawn 96 times (W(M96)); the second drawing replicates each pixel drawn 30 times (W(M30)). The drawing directions are set by the same offset directions:

V642446064600206

Table 7-1. V Command Functions

Function	Position Command	Comments
Draw a dot at the current location.	V[]	
Draw a line from the current location to location specified by [X-loc,Y-loc].	V[X-loc,Y-loc]	
Draw a line from the current location in the specified direction; use current multiplier. (Multiplier is specified via the W command.)	V *direction*	*direction* can be any of the following: 
Set temporary writing controls.	V(W(*writing options*))	See description of W (writing) command options.
Save current location on stack.	V(B)	
Restore most recently entered location from stack.	V(E)	

DRAWING CURVES

The C (curve) command provides options for drawing varieties of curves:

- Circles
- Arcs
- Open curves
- Closed curves

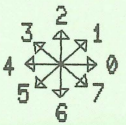
The C command provides the means for specifying the begin and end points of curves and circles and for specifying the cursor location after the writing operation for a curve. Table 7-2 lists the functions of the ReGIS C (curve) command.

Table 7-2. C Command Summary

Function	Curve Command	Comments
Draw a circle with radius equal to the distance between [X-loc,Y-loc] and current location.	C[X-loc,Y-loc]...	Leave current location at center.
Draw a circle with radius equal to the distance between [X-loc,Y-loc] and current location.	C(C)[X-loc,Y-loc]...	Leave current location at last location drawn.
Draw an arc of the length specified by *degrees*; radius of arc equal to the distance between the current location and [X-loc,Y-loc].	C(A *degrees*) [X-loc,Y-loc]...	Leave the cursor at the center.
Draw an arc of the length specified by *degrees*; radius of arc is equal to distance between the current location and [X-loc,Y-loc].	C(A *degrees*C) [X-loc,Y-loc]...	Leave the cursor at the last location drawn.
Interpolate a closed curve using the locations specified.	C(B)[] [X-loc,Y-loc]...[](E)	[] directs GIGI to draw at the current location.
Interpolate a closed curve using the specified offsets.	C(B)*direction*...(E)	*direction* can be any of the following: 
Interpolate an open curve using the locations specified.	C(S) [][X-loc,Y-loc]...[](E)	[] directs GIGI to draw at the current location.

continued next page

Table 7-2. C Command Summary continued

Function	Curve Command	Comments
Interpolate an open curve using the directions specified.	C(S)*direction*...(E)	*direction* can be any of the following: 
Set temporary writing controls for extent of Curve command.		C(W(*writing options*)) See description of W (writing) command options.
Interpolate a closed curve using the locations specified.	C(B)[][X-loc,Y-loc]...[](E)	[] directs GIGI to draw at the current location.
Interpolate a closed curve using the specified directions.	C(B)*directions*...(E)	*direction* can be any of the following: 

GIGI interpolates curves; that is, given a sequence of locations, GIGI computes intermediate points and connects them with small vectors which fit an appropriate curve. The shape of the curve depends on the locations specified in the curve command string, as shown in Figure 7-3.

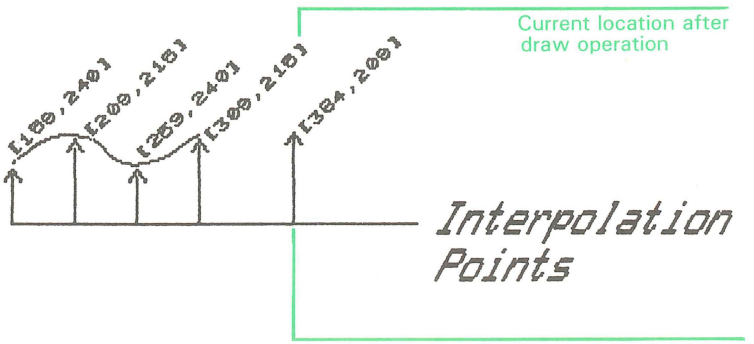


Figure 7-3. Example of Screen Locations GIGI Uses to Interpolate a Curve

Circles and Arcs

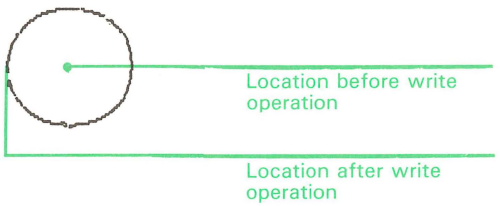
For circles and arcs, GIGI uses the location you specify to calculate the radius of the circle. Depending on the form of the curve command you use, the current location remains at the center of the circle or on the perimeter of the circle. Figure 7-4 shows the form of the C command which draws a circle and leaves the current location at the center. Figure 7-5 shows the form of the C command which draws a circle and leaves the current location on the perimeter of the circle.

Figure 7-4.
C Command Which Leaves Current Location at Center of Circle

```
ReGIS Command:
 p[350,250]
 c[+50]
```

Figure 7-5.
C Command Which Leaves Current Location at Perimeter of Circle

```
ReGIS Command:
 p[250,250]
 c(c)[+50]
```

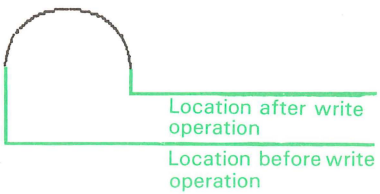


The same is true for arcs. Depending of the form of the curve command, GIGI leaves the current location either at the center of the arc or on the arc perimeter, as shown in Figure 7-6.

Figure 7-6.
Example of Arc Option of C Command

```
ReGIS Command:
 p[250,250]
 c(A+180)[+50]
```


```
ReGIS Command:
 p[250,250]
 c(c,a-180)[+50]
```



Open and Closed curves

Open curves and closed curves are curves for which you explicitly specify the points GIGI uses for interpolation. The general form of the C command for open and closed curves is:

$$C \left\{ \begin{matrix} S \\ B \end{matrix} \right\} [X\text{-loc},Y\text{-loc}][X\text{-loc},Y\text{-loc}]\ldots\ (E)$$

where (S) initiates an open curve and (B) initiates a closed curve. [X-loc,Y-loc] are points GIGI uses to interpolate the curve. (E) ends the interpolation sequence.

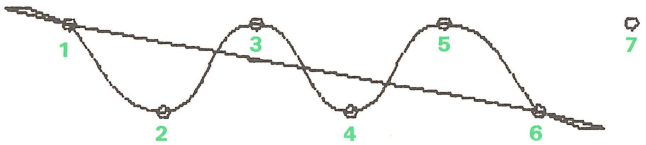
An open curve is a curve sequence that does not necessarily connect with itself, as shown in Figure 7-7.

Figure 7-7. Example of an Open Curve C Command

```
ReGIS Command:
c(s)[75,125]
    [150,200]
    [225,125]
    [300,200]
    [375,125]
    [450,200]
    [525,125](e)
```

A closed curve is one for which GIGI automatically closes the curve by connecting the line being drawn with the first location specified in the curve sequence. Figure 7-8 shows an example of how GIGI interpolates a closed curve.

Figure 7-8.
Example of a Closed
Curve C Command




```
ReGIS Command:
c(b)[75,125]
    [150,200]
    [225,125]
    [300,200]
    [375,125]
    [450,200]
    [525,125](e)
```

8 WRITING CONCEPTS

GIGI generates images in a variety of shapes, colors, and sizes. Basically, however, GIGI generates images on the screen by contrasting the color of the screen and the color of the writing pattern and by manipulating writing patterns and writing operations. This section describes the method GIGI uses to generate an image and describes some of the variations available for manipulating patterns.

The ReGIS W command provides controls for all of GIGI's writing functions. Table 8-1 lists the functions performed by the W command and provides the format for specifying the ReGIS command to perform the function.

THE WRITE OPERATION

A writing operation occurs each time GIGI interprets a V (vector), C (curve), or T (text) command. GIGI writes by changing the 0's that make up the screen background to 1's, which then become an image on the monitor screen. For example, when GIGI draws a vector from location [200,300] to location [204,296], the locations affected by that vector correspond to points [201,299], [202,298], [203,297], and [204,296] (a vector does not write its starting point).

The sequence of points GIGI passes through when drawing is referred to as a drawing path. The path comprises the first written point through the last written point for any writing operation.

Table 8-1. W Command Summary

Function	Writing Command	Comments
Set writing intensity	W(I(D)) or W(I0) W(I(B)) or W(I1) W(I(R)) or W(I2) W(I(M)) or W(I3) W(I(G)) or W(I4) W(I(C)) or W(I5) W(I(Y)) or W(I6) W(I(W)) or W(I7)	Dark Blue Red Magenta Green Cyan Yellow White
Set alternate (blink) Set no alternate	W(A1) W(A0)	
Set negative writing Set normal writing	W(N1) W(N0)	
Set shading on Set shading off Set character to shade with Set shading reference	W(S1) W(S0) W(S'*char*') W(S[,*Y-loc*]	 *char* is any character. *Y-loc* is any Y-coordinate.
Set writing pattern binary pattern pattern multiplier	W(*number*) W(P*number*) W(P(M*number*))	*number* is 1 through 6; defined by GIGI. *number* = binary number defining pattern. *number* = multiply pattern by a number 2 through 16.
Set offset length multiplier	W(M*number*)	*number* is number of times each pixels replicated for an offset V or P command.
Erase while writing Overlay while writing Complement while writing Replace screen contents	W(E) W(V) W(C) W(R)	

WRITING ATTRIBUTES

GIGI provides the means to assign attributes to each pixel drawn. You can assign the following attributes:

- Writing pattern appearance (for example, dots, dashes, solid lines)
- Area shading
- Writing color or intensity
- Negative writing (reversal of background and writing colors)
- Writing modes
- Temporary writing controls
- Alternation (the ability to make the line, curve, or text blink)

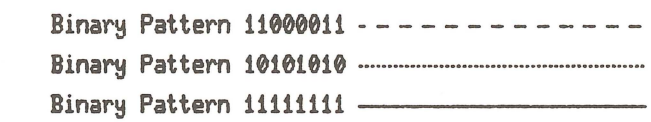
The following sections describe these attributes and how to use them.

Initial Writing Attributes

Initially, GIGI sets reverse video off (SET-UP RV0), writing pattern is pattern 1 (a solid line pattern), writing color is white, and writing mode is overlay. These are the initial settings GIGI uses when it starts up or whenever SHIFT-RESET is pressed. Using initial settings, GIGI writes a vector by setting the image memory bits corresponding to each point on the vector (except the starting point) to a 1 value. Also, the attribute memory attributes corresponding to each point on the vector are set to white.

Writing Patterns

GIGI draws lines and curves using a writing pattern. The writing pattern is a pattern of up to eight 1's and 0's. 1's set pixels (pixels are also called dots); 0's do not set pixels and leave a space in the pattern. In a V command, for example, GIGI uses the writing pattern when it draws the line. The example patterns below show how these settings affect the appearance of the writing pattern:

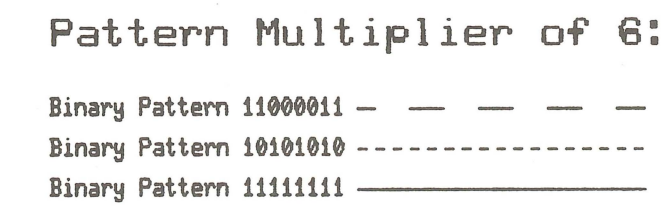


The first pattern writes dashes because GIGI sets only the pixels corresponding to the 1's during the write operation. The second pattern appears as dots separated by spaces because every other pixel is set. The third pattern is a solid pattern because all the pixels are set. You can direct GIGI to set dots in a specific pattern by means of the W command P (pattern) option.

Writing Pattern Multiplier

GIGI uses a pattern multiplier to multiply each bit in the pattern before that bit is displayed as a line on the screen. The power-up multiplier is 2; you can set the multiplier to values from 1 through 16.

The example patterns show how a pattern multiplier of 6 affects the appearance of the writing pattern:



This pattern writes dots and the spaces between them three times as long as the pattern in the previous example. This is because GIGI is multiplying each bit by the number of pixels set by the pattern multiplier (6, in this case) before it writes the pattern.

Area Shading

By means of the S (shading) option, GIGI allows you to shade an area as the writing operation is performed. When shading is in effect, the vector and curve commands follow the path from starting point to ending point as they would normally. However, GIGI automatically does additional writing to the video memory. GIGI shades using the current shading reference line and the current writing or text pattern, as explained below.

The Shading Reference Line

Starting with each point on the vector or curve path, a line is drawn to the shading reference line. The shading reference line is the horizontal line whose Y-coordinate is specified when shading is enabled. By default, the shading reference is the current Y-coordinate, but you can specify an explicit shading reference line. Figures 8-1 and 8-2 show how GIGI shades using the current Y-coordinate as the reference line.

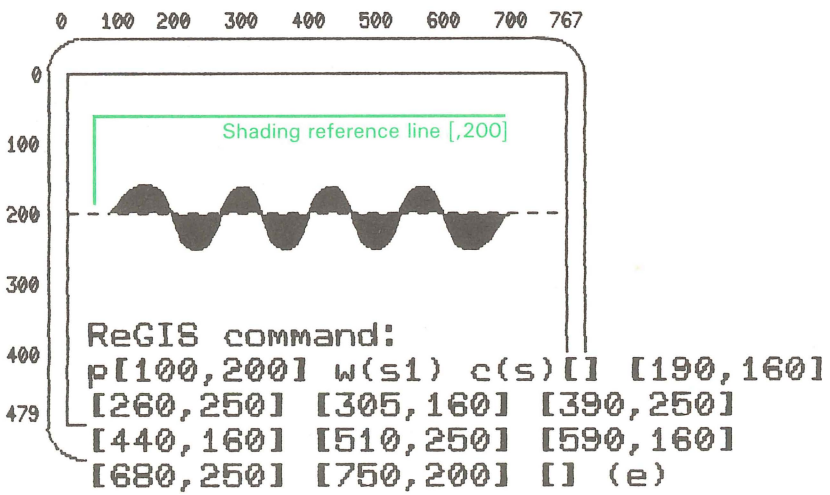


Figure 8-1. Area Shading with Shading Reference Set to [,200]

As you can see in Figure 8-1, the shading reference line is set to the current location [,200]. GIGI executes the curve command shown in the figure and shades the area between the reference line and the line being drawn. In Figure 8-2, below, the same curve command is executed but the shading reference is set to [,300] by the explicit reference.

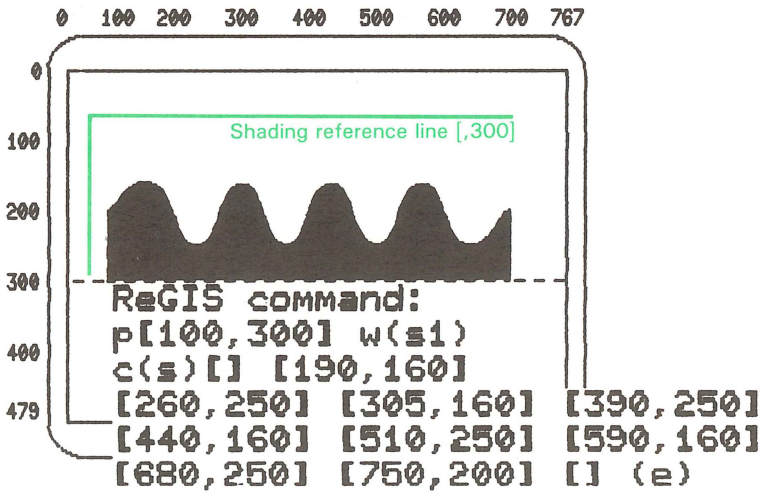


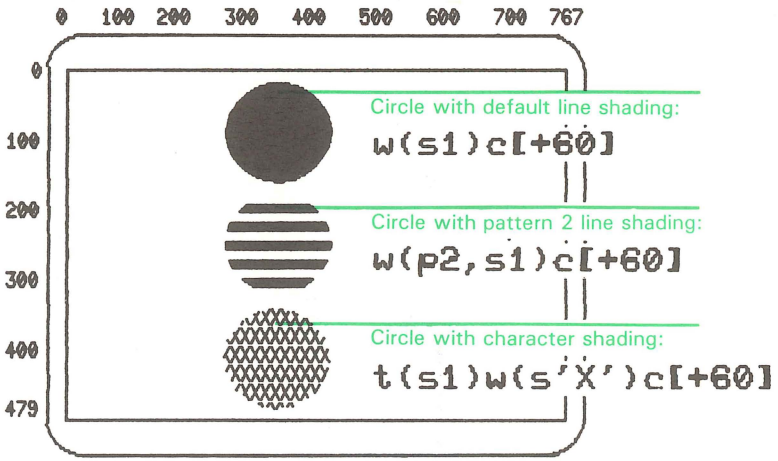
Figure 8-2.
Area Shading with
Shading Reference
Set to [,300]

Shading Patterns

GIGI shades using a shading pattern; this pattern is drawn on the vertical axis between the line being drawn and the current shading reference line, as shown in Figure 8-2.

This shading pattern can be either a line pattern or a text pattern. Line patterns are any of the patterns described in the section *Writing Patterns*, above. By default, the current writing pattern is used when writing the shading lines; in the examples above the current writing pattern is the default pattern of all 1's. You can also specify a text character from any of GIGI's character sets to be used as the shading pattern. Figure 8-3 shows examples of shading patterns.

Figure 8-3.
Examples of Line and Text Shading Patterns



Such shading can be used for area fill, but GIGI cannot automatically ensure that arbitrary areas are completely and correctly filled; you must ensure that the correct reference line or lines are specified to achieve the desired fill effect.

Creating the Effect of Area Fill

Because shading is based on the shading reference, you should shade drawings using the widest part of a picture, then erase (by writing in erase mode) the parts of the drawing which should not be shaded. (Writing modes are discussed in detail below.) Figure 8-4 shows how GIGI shades using the widest part of a shape and how writing in erase mode compensates for that.

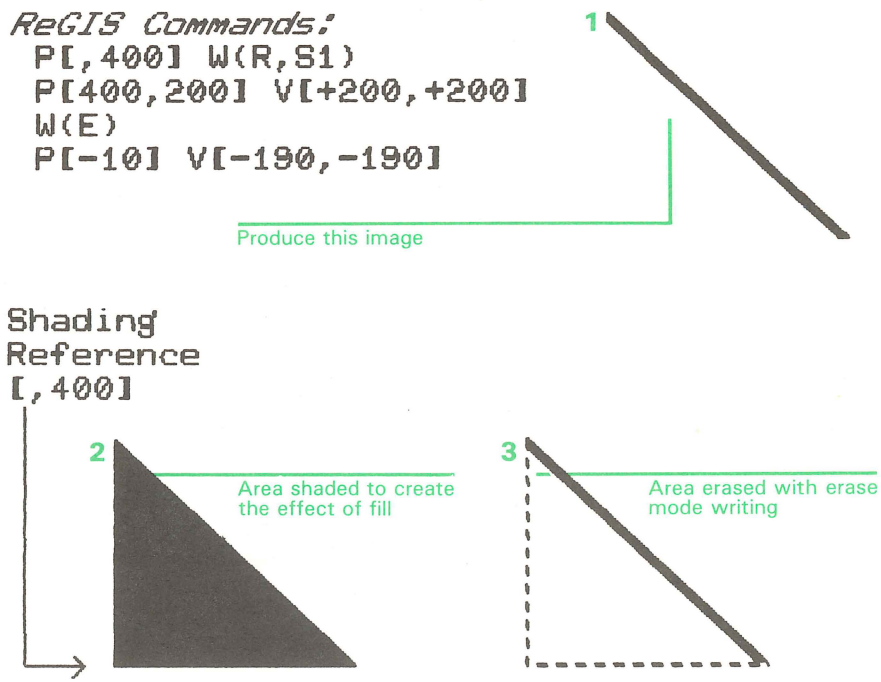


Figure 8-4.
Using Erase Mode to
Create Shading Effects

The first line of ReGIS code above sets the shading reference to [,400] and enables shading, W(R,S1). The second line of code draws a single line from [400,200] to the location 200 down and 200 to the right. However, because shading is enabled, GIGI appears to be drawing a triangular shape. Line 3 of the code enables erase mode writing and line 4 draws another single line but, while drawing in erase mode, GIGI erases the part of the triangle which was previously shaded.

Writing Color

GIGI allows you to use either a color or a monochrome (black and white) monitor and to specify the color or shade of gray used for drawing. There are three methods for specifying writing color:

- Intensity number
- Color mnemonic
- HLS (hue, lightness, saturation) specification

Each of these methods of specification corresponds to the others. Along with dark and white, GIGI lets you specify six colors:

Intensity Number	Color Mnemonic	Color
0	(D)	Dark
1	(B)	Blue
2	(R)	Red
3	(M)	Magenta
4	(G)	Green
5	(C)	Cyan
6	(Y)	Yellow
7	(W)	White

You can use the HLS specifiers rather than the number or letter method, but GIGI supports only the hue specification for the six colors listed above and dark and white. GIGI provides HLS for transportation from devices capable of supporting color variations with greater degrees of light intensity or grayness.

Intensity by Dot Spacing

By setting various writing patterns, i.e. the dot patterns described above, you can create the effect of darkening or lightening a color, as shown in Figure 8-5.

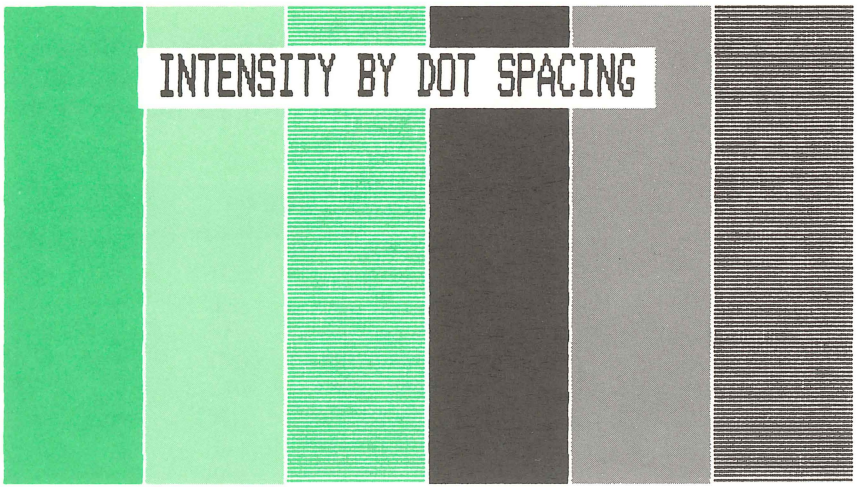


Figure 8-5. Varying Shades of a Color Using Shading Patterns

Negative Writing

ReGIS provides the means to direct GIGI to reverse the settings of bits in the bit map. Use the W command N option to set negative writing and direct GIGI to display using reverse video. Figure 8-6 shows examples of screens written in normal video (screen A) and reverse video (screen B).

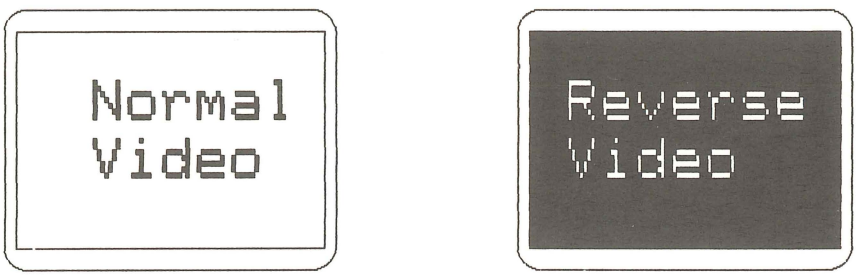


Figure 8-6. Example of Effect of Negative Writing

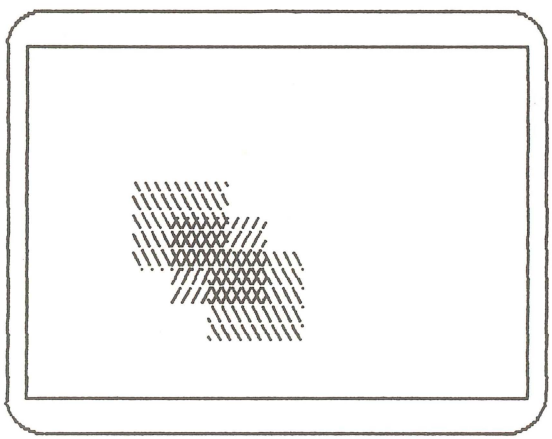
Writing Mode Options

GIGI provides writing options for performing logical operations with writing pattern dots and the dots that comprise the screen. These writing options provide the means for changing the settings of pixels in image memory. Initially, GIGI's screen is cleared and all the image memory bits are set to 0 (zero). As information is added to the display by writing, bits are set to 1. Using the pattern option, you can set GIGI to write in one of four modes:

- Overlay (V option)
- Replace (R option)
- Erase (E option)
- Complement (C option)

Overlay Writing

Overlay writing directs GIGI to overlay new images on top of images already on the screen. Figure 8-7 shows three boxes, each shaded with a different pattern to show how one pattern overlays another when the writing mode is overlay writing. Notice that the patterns that already appear on the screen are overlaid, not erased and rewritten.



In this mode,GIGI changes a pixel bit to 1 if the writing pattern bit is 1 and does not change the pixel bit if the writing pattern bit is 0. Thus if a pixel bit is 1, it stays 1; if a bit is 0, it is changed to 1 only if the writing pattern bit is 1.

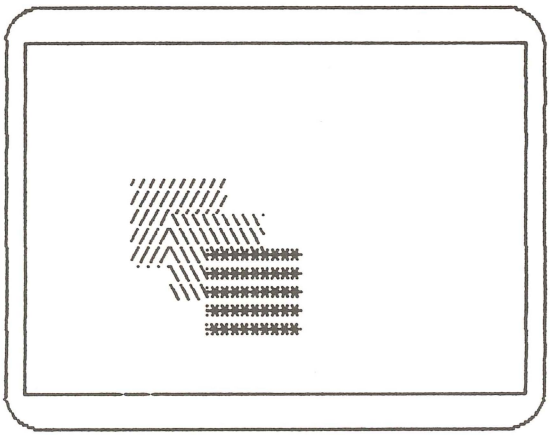
Overlay writing is GIGI's initial setting when powered on.

Replace Writing

Replace writing directs GIGI to replace images existing on the screen as new images are drawn.

Figure 8-8 shows the difference between overlay writing and replace writing. In overlay writing, the image appearing on the screen is not erased, it is simply overwritten. In replace writing, the image appearing on the screen is erased and replaced by the writing operation. You can see in Figure 8-8 that one pattern overlays the next and the previous image is no longer visible after the write operation.

Figure 8-8.
Example of Replace Writing

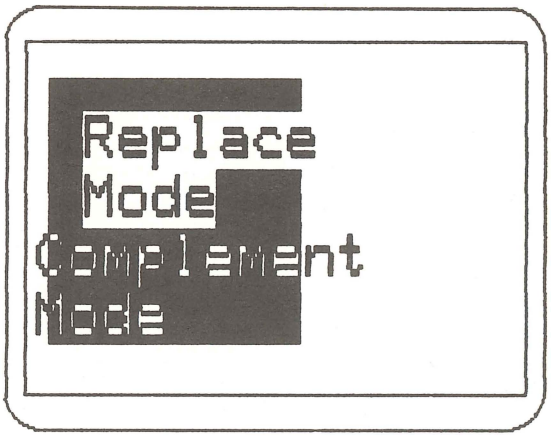


Replace writing forces the pixel bit to be the same as the pattern bit (either 0 or 1) regardless of the current value of the pixel bit.

Complement Writing

Complement writing directs GIGI to complement the existing image as new images overwrite existing image; that is, bits which are set are turned off and bits which are not set are turned on. In this example, two lines are written through a shaded box (bits are set to 1). The first line is written in replace mode, the second in complement mode.

Figure 8-9.
Example of Complement Writing



In complement writing, if the pattern bit is 0, GIGI does not change the pixel bit being written. But if the pattern bit is 1, GIGI changes the bit being written to 1 if it is 0, or to 0 if it is 1.

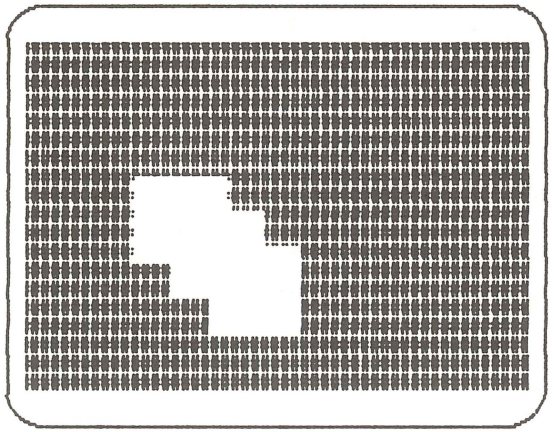
Complement writing is often performed twice in succession, where the first write operation makes an image appear and the second write operation erases the image without altering any other screen images.

Erase Writing

Erase writing directs GIGI to erase and not replace existing images as the write operation takes place. In erase writing, the writing pattern is not used. Normally, erase mode forces each bit written to be 0 unless negative writing is set. If negative writing is set, each bit written is forced to 1.

In the Figure 8-10 the entire screen is shaded before the erase writing operation and three boxes are drawn with erase writing in effect.

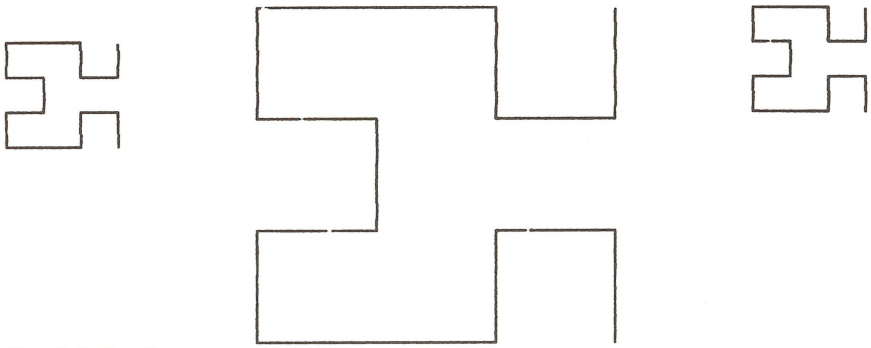
Figure 8-10.
Example of Erase Writing



Temporary Writing Controls

You can specify writing controls within a V, C, or T command; these options control attributes only for the extent of the command in which they are specified. For example, if the pixel multiplier is set to 30 in the W command, W(M30), GIGI uses 30 as the multiplier for all following commands. However, if you specify a temporary multiplier, the V command for example, that multiplier holds only for the extent of that command. Figure 8-11 shows and example of the use of temporary writing controls.

Figure 8-11.
Example of Temporary
Writing Controls



ReGIS Commands:

```
p[100,30]
W(M30)
V64244606460020 6
p[+400,30]
V(w(m96))64244606460020 6
p[+200,30]
V64244606460020 6
```

9 GIGI TEXT CONCEPTS

In graphics mode, GIGI displays characters by means of the T (text) command; you enter the characters to be displayed in a quoted string, for example:

T'SANDPIPERS'

You control the appearance of the characters in the string by specifying text options on the command line. These options set attributes for the characters in the string. Figure 9-1 shows examples of text GIGI can generate by means of text options:

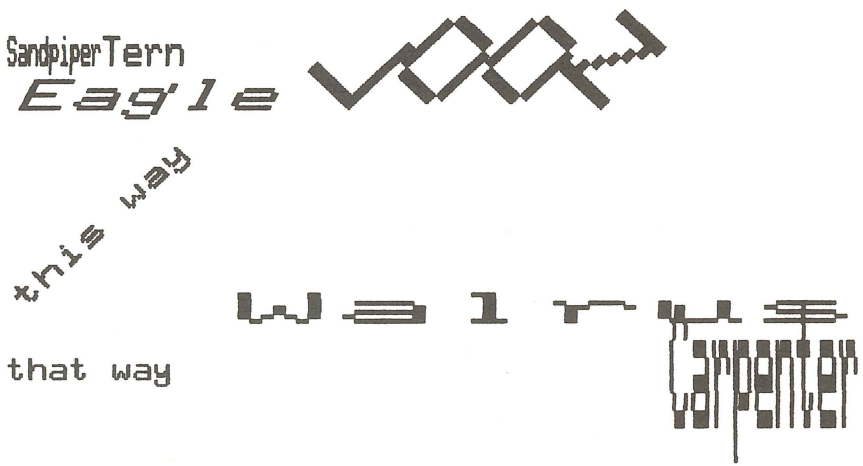
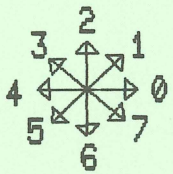


Figure 9-1.
Examples of GIGI's Text Attributes

The table below summarizes the T command functions and the options that control them; this chapter describes how to use these text options to control the appearance of a quoted string of text.

Table 9-1. Summary of T (Text) Command Functions

Function	Text Command	Comments
Set text size	T(S *number*)	*number* = 0 thru 16
Set text direction	T(D *angle*)	*angle* = 0 thru +360 or -360 (resolved to nearest 45 degree multiple)
Set text height	T(H *number*)	*number* = 1 thru 16
Set degrees of italic	T(I *angle*)	*angle* = 0 thru +45 or -45
Move text cell in specified direction	T *direction*	*direction* can be one of the following: 
Set temporary writing controls	T(W(*writing options*))	See writing command options.
Write text using characters from a specified character set	T(A *charset*)	*charset* = 0 thru 3
Save current text attributes and begin temporary attributes	T(B)	
Restore saved text attributes and end temporary attributes	T(E)	
Set explicit character size	T(S[*width,height*])	*width* and *height* are in pixels.
Set multiplier for explicit width and height	T(M[*mult,mult*])	*mult* = pixel; multiplier for width and height, respectively.
Set explicit character spacing	T[{±}*X spacing*, {±}*Y spacing*]	

This chapter first discusses the easily-used options of the text command:

- Character size
- Height
- Direction
- Italics
- Programmable character sets
- Temporary writing controls and attributes

The chapter then discusses how GIGI's text algorithm works and how to override the automatic options with explicit controls.

GIGI-CALCULATED TEXT OPTIONS

GIGI calculates four basic attributes for text characters:

- Size (width, height, and spacing of characters)
- Height (height of characters, that is short or tall)
- Direction (degree of tilt for the entire character display frame along a horizontal line)
- Italic (degree of slant in relationship to the character baseline)

The form for specifying these basic options is

(*keyletter value*)

where *keyletter* is a single letter that specifies which attribute to set and *value* assigns an appropriate numeric value to the parameter.

You can combine these options to produce various effects with text strings. In the examples in the following sections, all the options used to provide an effect are shown along with the displayed example.

Character Size

GIGI automatically calculates the size (width and height) of the text character and spacing between characters when you enter the Size option of the text command. The S option provides 17 character sizes (from 0 through 16). With the exception of size 0, GIGI generates characters with a width of 9 pixels and a height of 10 pixels. Figure 9-2 shows examples of the sizes you can specify using the size option.

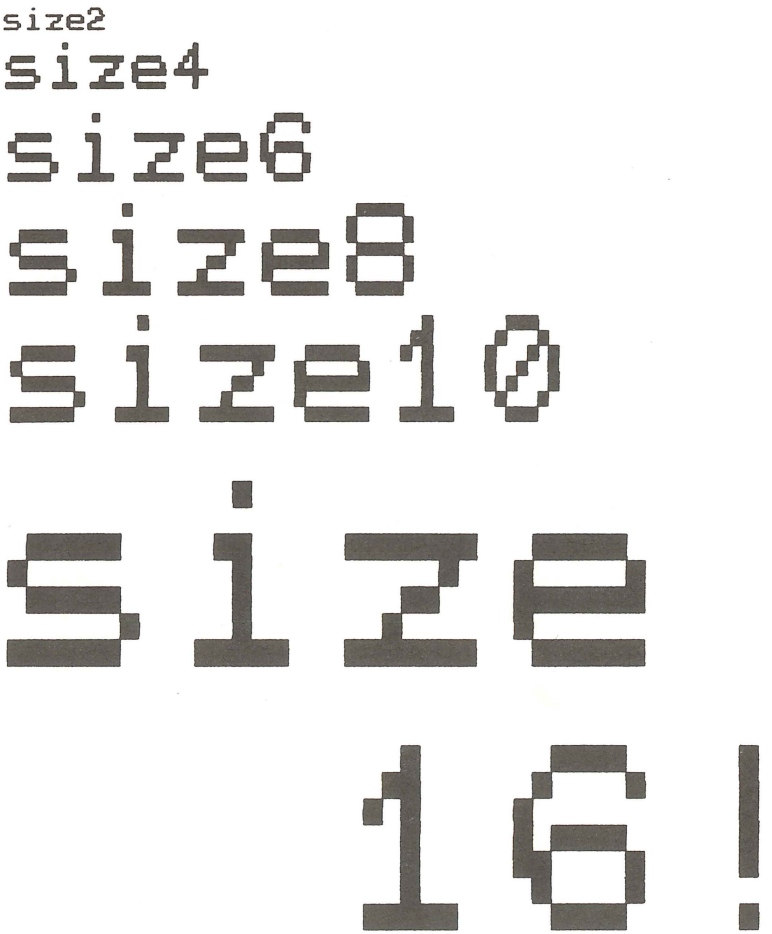


Figure 9-2.
Examples of Text Size
GIGI Generates
Automatically

These sizes are generated by the following type of command:

```
T(S0)'size0'
T(S2)'size2'
T(S4)'size4'
   .
   .
   .
T(S16)'size
  16!'
```

Notice that the spaces preceding size 16 are size 16 spaces.

Height

Rather than use the height established by the Size option, you can set the height of a character explicitly by entering the H (height) option. The H option must be specified after the S option is used to determine the width. You can specify values 1 through 16 for height. GIGI multiplies the value you enter for height times the height in pixels of the base character, 10. Figure 9-3 shows examples of the heights you can specify with the height option.

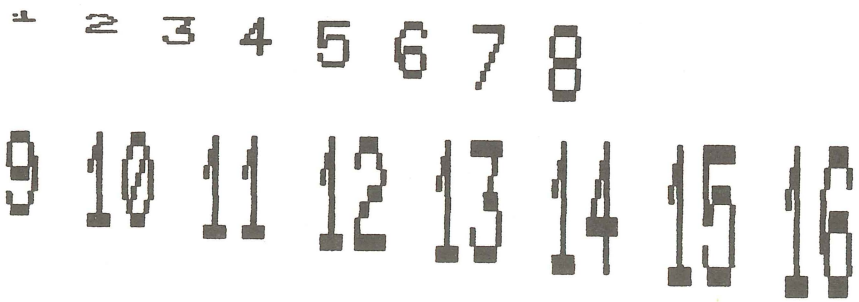


Figure 9-3.
Examples of Text Height
Generated the H Option

These sizes are generated by the following type of command:

```
T(H1)'1'
T(H2)'2'
   .
   .
   .
```

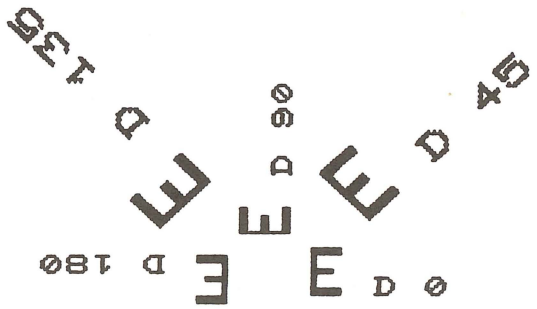
By combining the S and H options, you can generate various effects, such as short fat characters or tall skinny characters, as shown in Figure 9-4.

Figure 9-4.
Characters Generated by Combining the S and H Options

Direction

The direction option causes the characters in a quoted string to tilt a specified number of degrees on a horizontal axis. This option takes values in multiples of 45 degrees from 0 through 360; values can be either positive or negative. Figure 9-5 shows an example of how the direction option works:

Figure 9-5.
Examples of Text
Direction Option



There are cases where the individual characters tilt and where an entire string tilts. The order in which you specify the two options control which entity GIGI tilts.

To tilt individual characters in a string, set row direction with D, set size with S number, then set character direction with D angle, as shown in Figure 9-6.

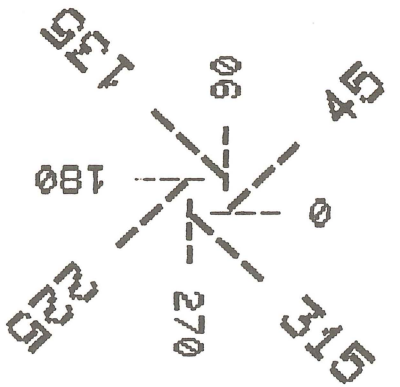
Figure 9-6.
Tilting Individual Characters

ReGIS Commands:

```
T(s3)(d0)  'DIRECTION 0'
T(D45) 'D I R E C T I O N 45'
T(D90) 'D I R E C T I O N 90'
```

To tilt an entire string, set direction for both string and characters with D angle, then set size with Snumber, as shown in Figure 9-7.

Figure 9-7.
Tilting an Entire String



ReGIS Commands:

```
T(D0)(S2)(d0)   '--- 0'
T(D45)(S2)(D45) '--- 45'
T(D90)(S2)(D90) '--- 90'
T(D135)(S2)(D135) '--- 135'
.
.
.
```

Italics

You can direct GIGI to slant characters in a quoted string a specified number of degrees forward or backward by entering the I option. Figure 9-8 shows examples of the various ways GIGI can display italics.

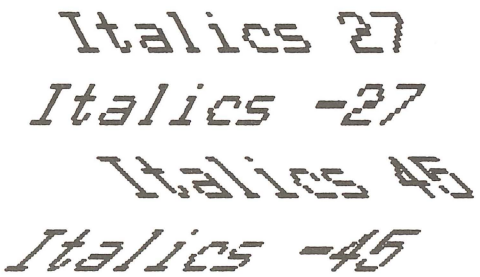


Figure 9-8.
Examples of Text
in Italics

Initial Text Attributes

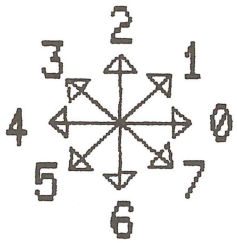
By default, GIGI uses the automatically-calculated options described above. When GIGI powers on or you press SHIFT/RESET, GIGI assigns initial attributes to text. Initial text attributes are:

S1
H2
D0
I0

When you use the automatic options to modify these default attributes, GIGI controls the process by which these options interact. You can control the effect one option has on others by overriding them with explicit display parameters. These explicit display parameters are discussed in the section *Overriding Automatically-Calculated Text Options*.

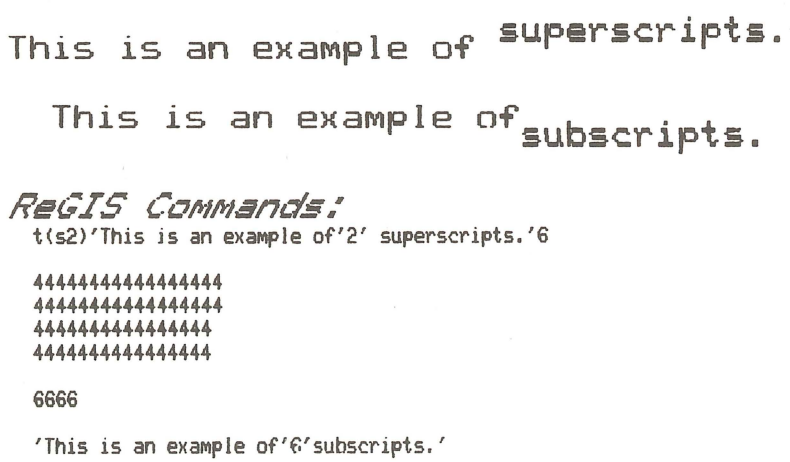
MOVING TEXT WITH OFFSET DIRECTIONS

GIGI's offset direction feature allows you to specify one of eight general directions:



These directions can be specified as options in a T command. For example, you can achieve the effect of subscripts and superscripts by moving the beginning writing position with offsets, as shown in Figure 9-9.

Figure 9-9. Example of Offset Direction Used to Display Text



In Figure 9-9, the offset direction 2 moves the text up to achieve the effect of a superscript. Directions 6 and 4 move the text writing position down and to the left; direction 6 also moves the text down to achieve the subscript effect.

SAVING AND RESTORING TEXT ATTRIBUTES

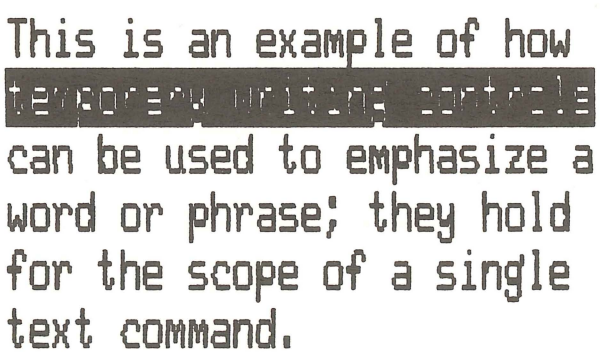
All Text attributes controlled by the T command can be saved as a unit and subsequently restored to their saved settings. The T command (B) option allows you to save the current attributes, make changes to one or more of the current attributes, then restore the saved attributes with the (E) option. The T command (E) option provides the advantage of restoring the saved attributes without requiring that you respecify them.

Only one set of attributes may be saved at a time.

TEMPORARY WRITING CONTROLS

Just as you can specify temporary text attributes by saving the current attributes and restoring them later, you can also specify temporary writing controls. For example, you may want to emphasize a single word in a sentence with color or reverse video; Figure 9-10 shows an example of temporary writing controls.

Figure 9-10. Example of How Temporary Writing Controls are Used



ReGIS Commands:

```
p[59,20]t'This is an example of how '
p[59,+50]t(w(n1))'temporary writing controls'
p[59,+50]t'can be used to emphasize a '
p[59,+50]t'word or phrase; they hold '
p[59,+50]t'for the scope of a single '
p[59,+50]t'text command.'
```

The W option allows you to nest the writing attribute for a single word or phrase, then return to the initial attributes for the remainder of the sentence. The remainder of the sentence must be specified in a new T command, because the temporary attributes remain in effect for the scope of the T command in which they are specified.

GIGI'S CHARACTER SETS

A character set is a group of characters you can display as text, for example the alphabetic, numeric, and special characters you see on a terminal keyboard. GIGI has four character sets, one of which is a native character set. This native character set consists of the 95 ASCII characters (listed in Table 1-1) you can display using GIGI's standard keypad.

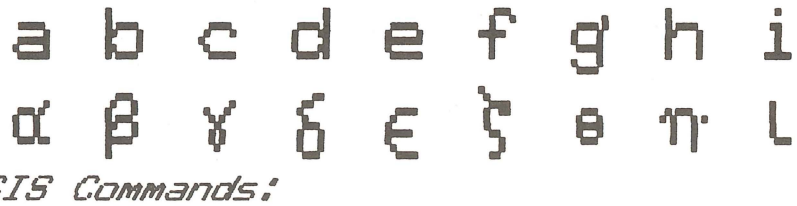
The remaining three character sets, referred to as alternate character sets, are programmable; you can create characters for these alternate character sets using the ReGIS L (load) command. You can define characters from another alphabet, for example the Greek alphabet, and use the Greek characters for text displays. In the example below, the first line is displayed using GIGI's native character set, the second line using an alternate Greek character set, and the third line using a character set designed for the APL programming language.

The ReGIS L (load) command provides the means for loading characters into the alternate character sets; you can then use the characters in text displays by means of the ReGIS T (text) command.

How GIGI Relates Alternate Characters to Native Characters

Each character in an alternate character set corresponds to a character in the native ASCII character set. This means that you can create characters that correspond to the positions for unshifted characters (for example, lowercase alphabetic characters), shifted characters (for example, uppercase alphabetic characters), and for the space character. This means that when you display the Greek characters you have defined in the a, b, and c positions of the Greek alternate character set, those characters correspond to the a, b, and c positions in the standard character set. Figure 9-11 shows two sets of characters typed with the same keystrokes, one using the native characters and the other with programmed Greek characters.

Figure 9-11.
Example of Text From Alternate Character Set



ReGIS Commands:

```
p[100,100]t(a0s4) 'a b c d e f g h i'
p[100,+70]t(a2s4) 'a b c d e f g h i'
```

This example assumes that the Greek character set is already loaded into character set 2. The sections below describe how to define and load character sets.

The L command allows you to specify a number and a name for each of the character sets and to define a text pattern for each of the characters comprising the character set.

Character Sets and Character Cells

Each of GIGI's character sets (including character set 0) consists of 95 character cells; each cell is 8 pixels wide and 10 pixels high. When used meaningfully, the cell contains an image such as an alphabetic character, for example, the letter T. In Figure 9-12, the cell on the left has no pixels set whereas the cell on the right shows how the letter T is formed by setting pixels.

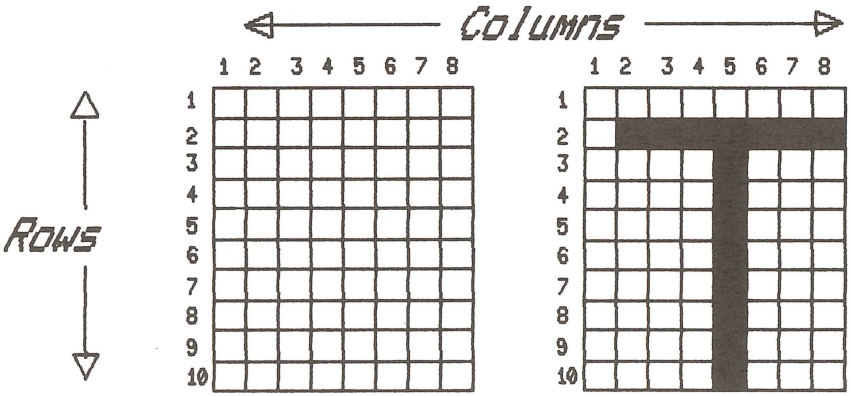


Figure 9-12.
Relationship of Text Cells and Characters

The image that comprises the letter T is created by setting pixels in the cell. When the letter is displayed, dots (i.e. the pixels that are set) are drawn and make the image in the cell look like the letter T. The image that you create when you set pixels in the character cell is called a text pattern.

Use the L (load) command, discussed in the paragraph below, to set the pattern.

Loading Characters into Text Cells

To load the cell, use two hexadecimal characters per width, for example FF.

The hexadecimal characters are 0 - 9 and A - F. Each hexadecimal character controls the setting for four pixels; two hexadecimal characters control the pixel setting for a single width of a cell. For example, FF sets all the pixels in a single width of a cell. Ten pairs of hexadecimal characters set all the widths in a character cell. For example, Figure 9-13 shows the hexadecimal pairs required to load the letter T into a character cell.

Figure 9-13. Example of Hexadecimal Digits Setting a Character Pattern

Hex Digit	Bits Set	Hex Digit	Hex Pair Per width
0	0 0 0 0 0 0 0 0	0	00
7	0 1 1 1 1 1 1 1	F	7F
0	0 0 0 0 1 0 0 0	8	08
0	0 0 0 0 1 0 0 0	8	08
0	0 0 0 0 1 0 0 0	8	08
0	0 0 0 0 1 0 0 0	8	08
0	0 0 0 0 1 0 0 0	8	08
0	0 0 0 0 1 0 0 0	8	08
0	0 0 0 0 0 0 0 0	0	00
0	0 0 0 0 0 0 0 0	0	00

GIGI uses this character cell as the reference unit when displaying various sizes of characters and when setting the proportions and spacing for characters. Based on ReGIS commands, GIGI creates a display frame for each character that is based on this reference unit.

Even though all ten rows are used when displaying characters as part of a text string, only the top eight rows are used when the character is specified as the shading character. Refer to the section *Area Shading* for information on shading characters.

When you specify the basic S option described above, GIGI propagates the first column of dot settings, which are always 0 in the native ASCII character set. When you use the explicit S option, you can directly control the propagation of the first row by specifying an explicit width value. This is especially important to remember when you are using alternate character sets.

The Load command provides the means for specifying which of GIGI's character sets you want to load or define. The Text command provides the means for specifying which of the character sets to use when displaying text.

OVERRIDING AUTOMATICALLY-CALCULATED TEXT OPTIONS

When you use the S, H, and D text options, GIGI calculates size, spacing, and direction for you. However, you can control the spacing of characters in a string as well as the size of characters by using explicit text display parameters. This section describes the method ReGIS uses to display characters and how you can use ReGIS processing along with explicit controls to display text strings.

Character Spacing

To create spacing for strings of characters, GIGI automatically generates a space as column 9 in the character frame. To set the pattern for column 9, GIGI replicates column 1 of the character. Thus, dots set in column 1 are replicated in column 9.

You can override this replication by explicitly specifying the number of columns GIGI is to display. The T command for overriding GIGI's automatic spacing is described in the section *Explicit Text Spacing*. Figure 9-14, below, illustrates how GIGI spaces between characters.

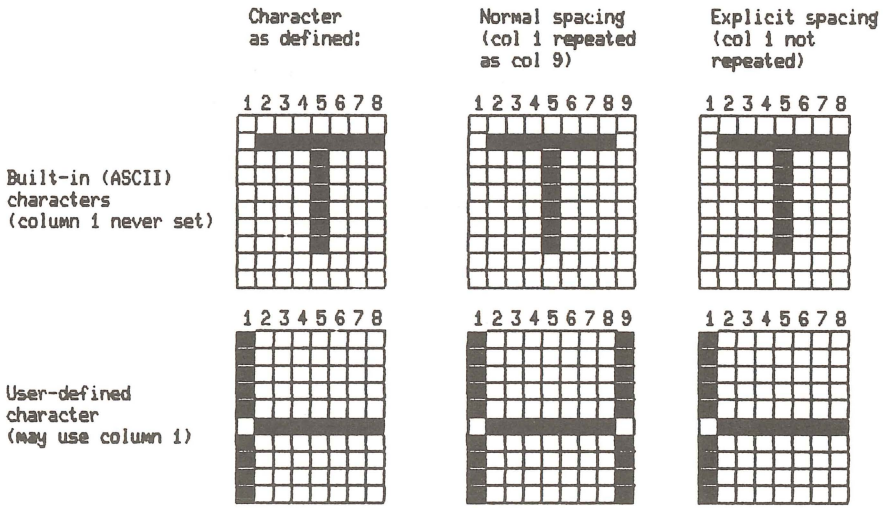


Figure 9-14. How GIGI Creates Spaces Between Characters in a Character String

In Figure 9-14, the character on the top row is from the native character set; the character in the second row is a user-defined character.

The left-most frames show how the characters are defined. For built-in ASCII characters, column 1 is never set because it is usually replicated. The user-defined character shows the dots set in column 1. The middle frames show GIGI replicating column 1 for the ASCII character and replicating the dots set in the user-defined character. The right-most frames demonstrate explicit spacing, in which GIGI does not replicate the first column.

How ReGIS Calculates Displays

GIGI displays text characters within a display frame. There is a single display frame for each character displayed. Using the frame for reference, GIGI displays characters in a quoted string in the following way:

- Determine current attributes and current text cursor location.
- Write a character in the current writing direction with size determined by sizing parameters (S[x,y]M[x,y])
- Update the current text cursor location using the spacing parameters.
- Get the next character in the quoted string and repeat the process beginning with step 1.

Remember that the basic text cell in GIGI's memory does not change; it is an 8-row by 10-column cell. Also, the character pattern is set in the cell. The following sections describe how to use these explicit controls.

Explicit Text Spacing and Direction

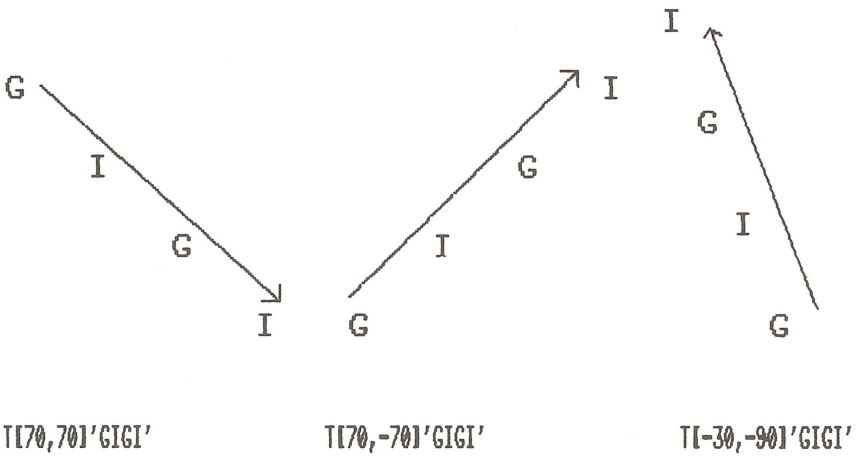
GIGI provides the means to explicitly set vertical and horizontal spacing for characters by specifying the text command in the form:

T[X-spacing,Y-spacing]

The sign (+ or -) for X-spacing sets the left-right character spacing; the sign of Y-spacing sets the up-down character spacing. Notice that this has the effect of setting the direction in which GIGI writes text.

Figure 9-15 shows examples of how to set spacing using explicit text parameters.

Figure 9-15. Examples of Explicit Spacing with T Command



The first example, ReGIS command T[70,70], spaces each new character 70 pixels across and 70 pixels down. The second example, ReGIS command T[70,-70], begins each new character 70 pixels across and writes up, using -70 as the vertical character spacing. The third example, T[-30,-90], shows GIGI writing backwards and upward.

Calculating Character Size

When you specify the S option, GIGI automatically calculates the character size using the following formula:

width = size X 9 pixels

height = (size X 1.5) X 10 pixels (rounded up)

You can explicitly override this calculation specifying size in the form:

T(S[*columns,rows*])

Columns specifies the number of columns the character occupies. *rows* specifies the number of rows the character occupies. The character size can be anything from 1 x 1 to 255 x 255.

GIGI needs one more piece of information, however, to go from the pattern in the 8 x 10 cell to the number of columns and rows you specify: the number times to write each column and row of the pattern before writing the next column or row of the pattern. This information comes from the multiplier M[*a,b*], where a is the column multiple and b is the row multiple. (That is,write each column pattern *a* times before writing the next column; write each row *b* times before writing the next row in the pattern).

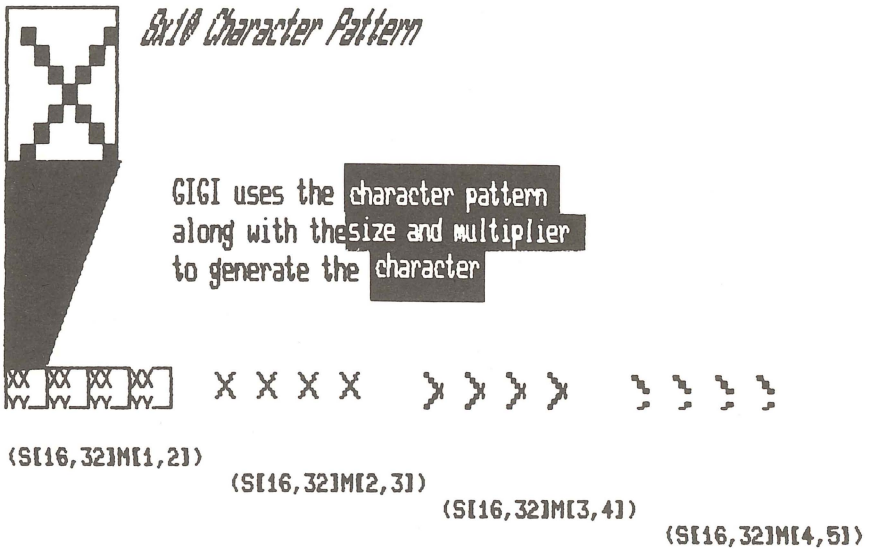
Figure 9-16 shows the string XXXX written four times. Both the spacing and the character size remain constant in the example:

```
T[30] ;'Space to the right 30 pixels for each character'
T(S[16,32]) ;'Base size is 16 columns, 32 rows per character'
```

The actual ReGIS commands are:

```
T[30](S[16,32]M[1,2])'XXXX'
T[30](S[16,32]M[2,3])'XXXX'
T[30](S[16,32]M[3,4])'XXXX'
T[30](S[16,32]M[4,5])'XXXX'
```

Figure 9-16.
Example of How ReGIS
Performs Explicit Spacing



The boxes are drawn in 30 pixel squares to show the proportion set by the spacing command, T[30]. The size option in each command specifies 16 columns and 32 rows, so that each character displayed is 16 by 32. As the multiplier value becomes greater, you can see that the width and depth of the columns and rows in the characters become proportionally greater. In the first frame, the character does not quite take up the entire space specified by the spacing command; in the last frame, only the top left portion of the character appears because there is room for only the first three rows and six columns to be replicated.

10 MACROGRAPHS

The ReGIS Macrograph is a facility for storing and recalling text strings.

Macrographs are ReGIS command strings (or any arbitrary string of characters) which you can insert in a ReGIS command stream by means of the macrograph command (@). When the macrograph is executed, its contents are inserted in-line at the position in the command stream at which it is executed.

Defining a Macrograph

Use the macrograph command to define, delimit, and invoke a macrograph definition text string. You define macrographs in the form:

@:*keyletter string* @;

GIGI interprets @: as the delimiter which precedes the macrograph string. GIGI interprets @; as the delimiter which terminates the macrograph definition.

Keyletter defines one of the 26 letters of the alphabet to be the name of the macrograph. GIGI interprets uppercase and lowercase characters as the same keyletter.

String specifies the character string that comprises the contents of the macrograph. *String* has no fixed maximum length. However, the number of characters used by macrographs and programmable key storage may be no more than 2000 characters.

Generally, *string* is a part of or a whole ReGIS command string that is used frequently.

Once the macrograph is defined and stored, you can invoke it anywhere in a ReGIS command stream, thereby directing GIGI to interpret the text string. The following section shows how to invoke and use macrographs.

INVOKING AND USING MACROGRAPHS

You use macrographs by invoking them in appropriate places in ReGIS command streams. You invoke the macrograph using the following form:

@keyletter

Keyletter specifies the name of the macrograph whose text string is to be inserted.

Macrographs may be defined and invoked anywhere in a ReGIS string except within quoted text.

Figure 10-1 shows a macrograph definition along with the shape GIGI displays when the macrograph executes.

MACROGRAPH STORAGE

You can store up to 26 macrographs in macrograph storage; macrograph definitions may as large as necessary, up to 255 characters. However, macrographs share a GIGI storage sector with programmed keypad definitions and that use of that sector is up to the programmer. Use the ReGIS R (report) command to determine the amount of space available for macrographs.

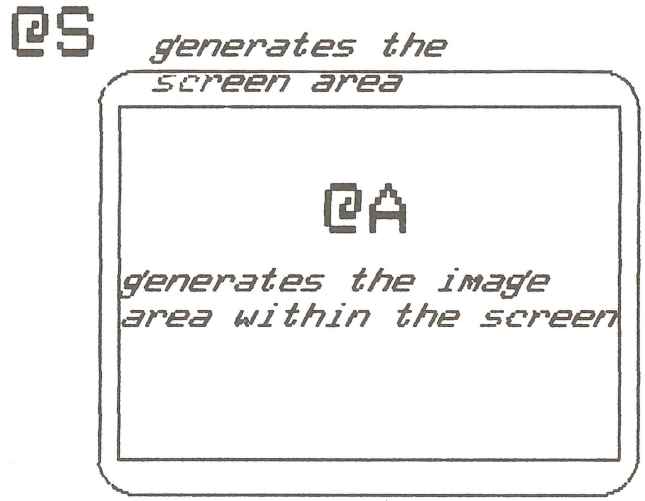
Clearing Macrograph Storage

Use the following form of the macrograph command to clear macrograph storage:

@.

The at-sign followed by the period clears all current macrograph definitions.

Figure 10-1.
Example of Macrograph Definition



Macrograph Definition:

```
s(e)
;'define the screen'
@:S p[100,135]
     c(c,a-90)[+30]v[+370,]
     c(c,a-90)[,+30]v[,+300]
     c(c,a-90)[-30]v[-370]
     c(c,a-90)[,-30]v[,-300]@;
;'define an image area'
@:A p[115,135]
     v[+400,][,+300][-400,][,-300]@;

;'invoke the macrographs'
@S
@A
```

11 HOW GIGI DISPLAYS IMAGES

This chapter contains information on how GIGI displays images and is meant for programmers who are familiar with GIGI's graphics functions. The chapter describes GIGI's video memory and ideas related to video memory.

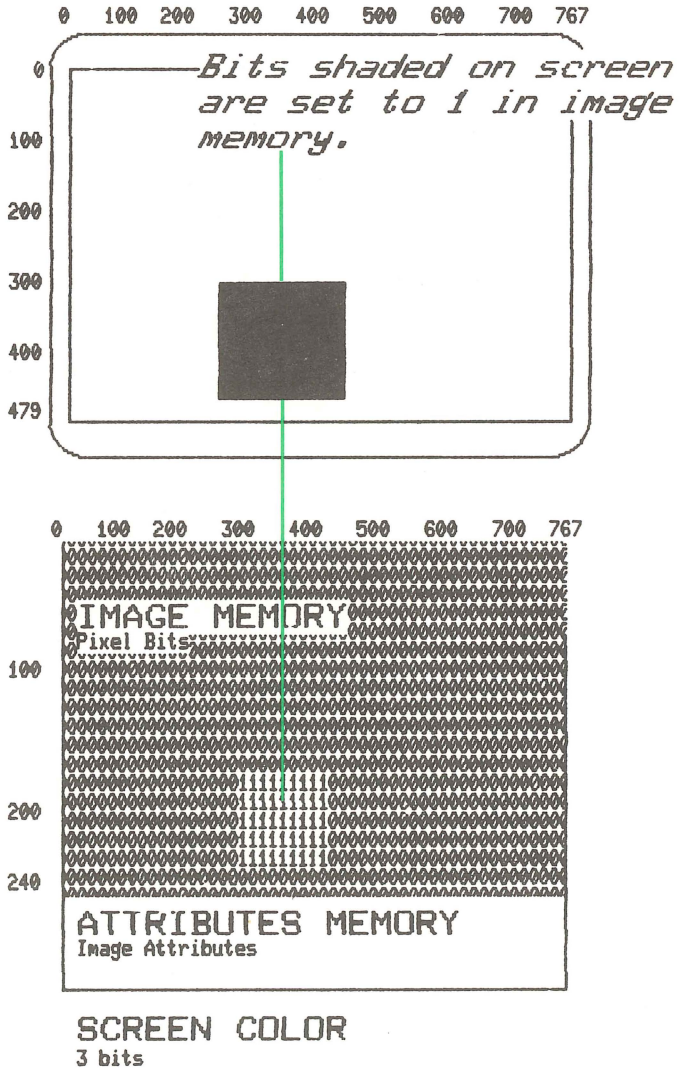
Pixels and Video Memory

GIGI keeps track of each even-line pixel on the screen using *video memory*. Video memory consists of image memory and attributes memory. *Image memory* consists of one bit of GIGI memory for each pixel on the screen; bits can be set to 1 or 0. That is, each bit in image memory corresponds to a pixel on the screen.

GIGI uses bits in image memory to determine whether to display pixels using screen attributes or image attributes. GIGI displays pixels whose bits are set to 0 using screen attributes; GIGI displays pixels whose bits are set to 1 using image attributes. GIGI maintains both screen and image attributes in attributes memory.

Figure 11-1 shows the layout of video memory and the relationship between memory and the screen.

Figure 11-1.
Screen Image Mapped to Image in GIGI's Video Memory



In Figure 11-1, the image draw on the screen maps to an image in video memory. You can see that bits set to 1 in image memory form a rectangle; GIGI uses the attributes in image attribute memory to display pixels corresponding to these bits. For example, the color attribute for images might be yellow. The other bits are 0's; GIGI uses the attributes set for screen attribute memory to display pixels corresponding to these bits.

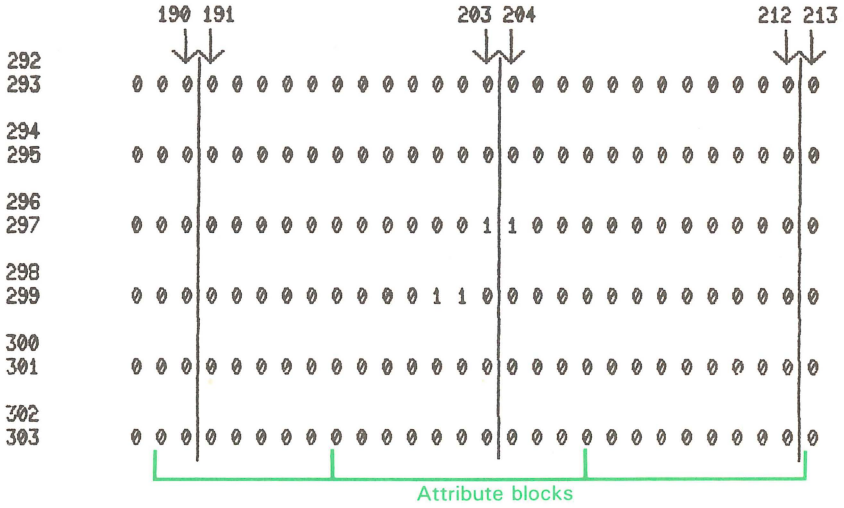
GIGI refreshes the monitor screen 50 or 60 times per second (depending on the power frequency set) using the contents of image memory and the attributes memory. This refresh speed allows the images to change rapidly when several commands are interpreted; the image does not change at all if there are no commands to process.

Attributes Memory and Attribute Blocks

Attribute memory consists of three color bits and one blink bit for each group of 12 horizontal even-line pixels, 64 groups per line. For example, pixels [5,200] and [10,200] have the same attributes when their image memory bits are set, whereas pixels [10,200] and [13,200] may have different attributes. This group of pixels with the same attributes set is seen most easily in reverse video.

This group of 12 pixels is referred to as an attribute block; attributes blocks in video memory affect the appearance of images drawn on the screen. For example, when GIGI draws a vector from location [200,300] to location [204,296], the locations affected by that vector correspond to points [201,299], [202,298], [203,297], and [204,296], as shown in Figure 11-2.

Figure 11-2.
Attributes Blocks and
Even-Odd Locations



These points correspond to four bits in image memory that are set to 1, but only three sets of attributes in attribute memory.

GIGI maps the first two points ([201,299], an odd Y-coordinate, and [202,298], the next lower even Y-coordinate) to the same line on the screen. Their X-coordinate values are within the same 12-pixel attribute block.

The third and fourth points ([203,297] and [204,296]) are both on the next higher distinguishable line; but their X-coordinate values lie on either side of a 12-pixel attribute block, and, therefore, may have different attributes.

Advantages of Divided Video Memory

Because video memory is comprised of a separate image memory and attribute memory, GIGI can change attributes in one memory without changing pixels in the other and vice versa. The following two sections describe how to take advantage of divided video memory.

Inhibiting Changes to Attributes Memory

Use the I (intensity) option to the W (writing) command with no argument (i.e., comma, right parenthesis, another letter after the I), to inhibit changes to attribute memory for subsequent writing operations.

This technique allows you to change image memory (i.e., draw images) at the same time locations on the screen retain whatever screen color (or brightness) is set for them.

Inhibiting Changes to Image Memory

You can change attribute memory without changing image memory by setting the desired color (or brightness) and blink and setting overlay writing mode with writing pattern 0. Subsequent writing operations make no change to image memory, but set the attribute sets for all points on their paths.

A writing operation occurs each time GIGI interprets a V or C command and an image appears on the screen. That image is drawn using the pattern you specify. For example, you can specify a pattern consisting of dots and dashes, a pattern consisting of only dots, or a pattern with all the dots set (the default).

Reverse Video

GIGI provides the ability to reverse the interpretation of attributes for the screen and for drawings and text. You can set reverse video using the RV set-up and using the N (negative background) option of the S (screen) command.

When you set reverse video, GIGI interprets the bits set in image memory in reverse. For example, if an image on the screen is blue and the screen color is red, GIGI reverses its interpretation of image bits for screen and writing. In this example, GIGI reverses the interpretation by interpreting the image as red and the screen color as blue.

12 ReGIS COMMAND DESCRIPTIONS

This chapter contains detailed descriptions and syntax rules for coding ReGIS commands. The following commands are listed in alphabetical order for easy reference:

- Curve
- Load
- Macrograph (@)
- Position
- Report
- Screen
- Text
- Vector
- Writing

Refer to Chapter 4, *Executing ReGIS Commands*, for information on how to execute ReGIS commands from the keyboard or by means of programs. Refer to Chapter 5, *Coding and Using ReGIS Commands*, for information on the syntax rules for ReGIS commands and information on how ReGIS interprets commands.

CURVE COMMAND

The C (curve) command performs generalized curve interpolation functions for drawing:

- Circles
- Arcs
- Open curves
- Closed curves

The following sections describe how to code these types of curves. Refer to Chapter 7, *Drawing With ReGIS,* for examples of how to draw curves and circles.

Circles

The format for specifying circles is:

$$C \left\{ \begin{array}{l} (C) \\ (W(\textit{temporary-writing-controls})) \end{array} \right\} \left\{ \begin{array}{l} [\textit{X-loc, Y-loc}] \\ \textit{direction} \end{array} \right\} \ldots$$

C[*X-loc, Y-loc*]...
Directs GIGI to draw a circle centered on the current location. [X-loc,Y-loc] may be specified as either an absolute or relative value.

This form of the curve command does not move the current location; when the command completes execution, the cursor and the current location remain at the center of the circle.

If an absolute location is entered, it specifies a point on the circumference; the radius of the circle GIGI draws is the distance between that location and the current cursor location. For example, when the cursor is located at screen location [200,200] and the command C[20,20] is entered, the radius of the circle GIGI draws is the distance between the absolute screen location [20,20] and the current cursor location.

C(C)[*X-loc, Y-loc*]
Directs GIGI to draw a circle with the center at [*X-loc, Y-loc*]; the current location is on the circumference. GIGI uses the distance between the current cursor location and the screen location specified by [*X-loc, Y-loc*] as the radius. The cursor and current location remain on the circumference after the writing operation.

W(*temporary-writing-controls*)
Enter temporary writing controls. *Temporary-writing-controls* are those listed and described in the description of the W (writing) command.

If a relative distance is entered, the radius is the distance from the current cursor location specified by the relative location. For example, when the cursor is located at location [200,200] and the command C[+ 20] is entered, GIGI draws a circle with a radius of 20 pixels, a smaller circle.

Cdirection
Directs GIGI to draw a circle centered on the current location; the radius of the circle is determined by the current pixel multiplier specified by the writing (W) command. For example, if the multiplier is 1, the radius of the circle is one pixel. If the multiplier is 10, the radius is ten pixels.

Arcs

The format for specifying arcs is:

C { (A *degrees*) / (C) / (W (*temporary-writing-controls*)) } { [*X-loc, Y-loc*] / *direction* } ...

C(A *degrees*) [*X-loc, Y-loc*] ...
Directs GIGI to draw an arc of the length specified by degrees. The center of the arc is the current location; the arc begins at [*X-loc, Y-loc*]. The radius of the arc is the length between the current location and the location specified by [*X-loc, Y-loc*]. GIGI draws the arc counterclockwise for a positive number of *degrees* and clockwise for a negative number of *degrees*.

C(A *degrees*) *direction*
Directs GIGI to draw an arc centered on the current location; the radius of the arc is determined by the current pixel multiplier specified by the writing (W) command. For example, if the multiplier is 1, the radius of the arc is one pixel. If the multiplier is 10, the radius is ten pixels.

C (A *degrees*C) [*X-loc, Y-loc*]
Directs GIGI to draw an arc with the center at [*X-loc, Y-loc*]; the arc begins at the current location. GIGI uses the distance between the current location and the location specified by [*X-loc, Y-loc*] as the radius. The cursor and current location remain at the end of the arc after the writing operation. GIGI draws the arc counterclockwise for a positive number of *degrees* and clockwise for a negative number of *degrees*.

W(*temporary-writing-controls*)
Enter temporary writing controls. *Temporary-writing-controls* are those listed and described in the description of the W (writing) command.

Closed Curves

The format for specifying a closed curve is:

C (B) *closed-curve-sequence* (E)

This form directs GIGI to draw a closed curve based on a sequence of locations specified by *closed-curve-sequence*. *Closed-curve-sequence* can be specified in the form:

[*X-loc, Y-loc*][*X-loc, Y-loc*]...

or

direction direction ...

which specify the locations or directions along which GIGI interpolates the curve. A sequence generally requires at least three explicit locations or directions to represent a useful curve.

The current location at the time (B) is processed is the first point used in calculating the curve. GIGI interpolates the curve through all points specified in the sequence. When the draw operation completes (after (E) is processed), the current location is at the second point specified in the sequence. For example,

```
P[200,200]
C(B)[220,220][240,200]
     [260,220][250,250]
     [230,300]
  (E)
```

The starting point of the curve sequence is [200,200] and GIGI interpolates the curve through all of the points specified in the sequence, leaving the current location at the second point specified, [240,200].

(E) specifies the location at which GIGI completes the interpolation.

Open Curves

Specify open curves in the following form:

C (S) *open-curve-sequence* (E)

This form directs GIGI to draw an open curve based on a sequence of locations specified by *open-curve-sequence*. *Open-curve-sequence* is specified in the form:

[*X-loc, Y-loc*] [*X-loc, Y-loc*] ...

or

direction direction...

These locations or directions define the points along which GIGI interpolates the curve. A sequence generally requires at least three explicit points to represent a useful curve.

The current location at the time the (S) is processed is the first point used in calculating the curve. GIGI then interpolates through all the points in the sequence, but draws a curve only through the next-to-last location specified in the sequence. The current location after GIGI completes the draw operation is the last point specified in the sequence.

The arc between the current location and the first point in the curve sequence is not drawn; however, the value of the current location determines the starting slope of the curve. You can include the current location in the arc by entering [ ] as the first point in the sequence.

Likewise, the arc between the last and next-to-last points in the curve sequence is not drawn and the value for the last point determines the slope of the last arc drawn. The last value in the sequence can be included in the sequence by following it with [ ].

To draw an approximation of an ellipse, specify a closed curve interpolation sequence, giving the end points of the major and minor axes as the four positions.

Terminating a C command while GIGI is processing an interpolation sequence leaves sequence open so that the curve sequence can be continued in a subsequent C command. Note that clearing the screen resets the curve sequence.

LOAD COMMAND

Each of GIGI's three alternate character sets consists of 95 cells, each cell of which corresponds to a single ASCII character.

Character set 0 is the built-in ASCII character set and cannot be altered. The three loadable character sets are selected by the character set designators for 1, 2, and 3. Refer to Chapter 9, *GIGI Text Concepts*, for information on how to use alternate character sets.

The L (Load) command performs the following functions:

- Defines the name or number of one of GIGI's three alternate character sets.
- Defines the shape (i.e. dot pattern) of the individual characters that comprise any of the three alternate character sets.
- Loads characters into any of the three alternate character sets.

The format for specifying the L command is:

L { (A*charset-number*) / (A'*charset-name*') } *'character' pattern*;

Where:

(A *charset-number*)
Specifies the number of the character set to load in this and subsequent L commands. You can specify 1, 2, or 3. If you are specifying both a number and a name for the character set, specify the number first.

(A'*charset-name*')
Define a 1- to 10-character name for the character set; select the character set for loading. If you are specifying both a number and a name for the character set, specify the number first.

'*character*'
Specifies a single character from the native ASCII character set to be associated with the dot pattern defined by *pattern*, below. The 95 loadable cells correspond to the ASCII characters space through tilde. These ASCII characters and their decimal equivalents are listed in Table 1-1.

If the string is multicharacter, only the first character is used and the remainder of the string is discarded.

pattern
Specifies hexadecimal digits 0 through F, that is, 0 through 9, A, B, C, D, E, F. (Both uppercase and lowercase characters may be used.) These digits translate to a dot pattern for the text cell specified by *character*, above.

Cells are always loaded using hexadecimal characters. One row at a time is loaded from the top row through the bottom row. Character cells are eight units wide and ten units high.

For usage information on how to load text cells, refer to Chapter 9, *GIGI Text Concepts*.

;
The semicolon character terminates the L command. The semicolon (;) is required to terminate the L command. This prevents ReGIS from misinterpreting a hexidecimal digit, for example, the character C could be misinterpreted as a C command.

Only one character cell may be loaded with a single L command.

The pattern selected for loading by the (A) option is independent of the one selected by the T command. Changing one does not affect the other.

MACROGRAPHS

Macrographs are command strings (or any arbitrary string of characters) which you can insert in a ReGIS command stream by means of the macrograph command (@).

The contents of a macrograph are inserted in-line at the position in the command stream at which the macrograph is invoked.

The forms for defining and invoking macrographs are:

Defining a Macrograph:
@:keyletter character-string @;

Invoking the Macrograph:
@keyletter

Where:

@:
Initiates definition of a macrograph and specifies that the following alphabetic keyletter is the name of a macrograph.

keyletter
Defines one of the 26 letters of the alphabet to be the name of the macrograph. GIGI interprets both uppercase and lowercase characters the same.

Macrographs may be defined anywhere in a ReGIS string except within quoted text.

When you define a macrograph which already exists, the previous contents are cleared before the new definition is saved.

A null definition is legal and clears an existing macrograph.

character-string
Specifies the character string that comprises the contents (extent) of the macrograph. *character-string* has no fixed maximum length. However, the number of characters used by macrographs and programmable key storage may not exceed 2000 characters.

Generally, *character-string* is a part of or a whole ReGIS command string that is used frequently or which has a special purpose such as the composition of a special mosaic character.

Neither the initiator (@:) nor the terminator (@;) are saved as part of character-string.

No ReGIS command processing is performed while the macrograph is being saved.

A macrograph may invoke other macrographs. All characters present between the initiator and the terminator, including all control characters, are saved in the macrograph definition.

@;
The commercial at-sign (@) and semicolon (;) terminate the macrograph definition.

@keyletter
Invokes the macrograph defined by *keyletter* and inserts the *character-string* currently associated with that macrograph. The macrograph contents are inserted at the point in the ReGIS command stream at which the macrograph is invoked.

Macrograph invocation may be nested to the limits of device memory, but may not be directly or indirectly recursive.

Macrographs may be invoked at any point in a ReGIS command stream, except within a quoted string. That is, T'@A' displays the characters @A whether or not A is defined as a macrograph.

A macrograph may not be used to supply an argument to an outstanding at-sign; that is, @@ is illegal.

Invoking an empty macrograph is not an error.

POSITION COMMAND

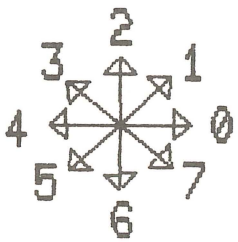
The P (Position) command sets or changes the current location and moves the graphics cursor to the new current location without writing to the screen. Refer to Chapter 9, *Screen Concepts*, for usage information on the P command. The format for specifying the P command is:

P { [*X-loc, Y-loc*] | *direction* | (W (M *multiplier*)) | (B) | (E) } ...

[*X-loc, Y-loc*]
Specifies a location on the screen to which the current location is moved.

You may specify either absolute or relative values for *X-loc* and *Y-loc*. For information on relative and absolute address values, refer to the section *Positioning the Cursor* in Chapter 6.

direction
Moves the cursor in the direction specified by direction; valid entries are 0 through 7. 0 through 7 move the cursor according to the following scheme:



(W(M *multiplier*))
For the scope of the current P command, sets temporary multiplier for the current writing pattern. This multiplier is useful in moving the current location with the *direction*, described above. Syntactically, you can specify any of options described in the W (writing) command; however, only the multiplier has an effect. For information on how to specify and use writing control options, refer to the description of the W command and to Chapter 8, *Writing Concepts*.

(B)
Directs ReGIS to save the current location. Up to eight locations may be saved on a stack.

(E)
Directs ReGIS to retrieve the most recently saved position from the position stack.

P(B) followed by positioning and drawing commands, then followed by P(E), executes the commands, then restores the current location to the location saved by the P(B) command.

REPORT COMMAND

The R (Report) command provides a mechanism for reporting various values to the host or to the monitor screen:

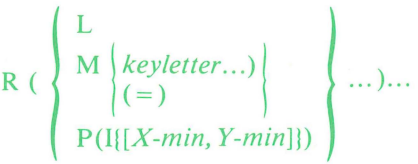
- The name of the current alphabet (the L option).
- The contents of the specified macrograph (the M option).
- The current screen location of the graphics cursor (the P option).

If multiple reports are requested in a single report command or report command option, each report returned is delimited (by a carriage return) as if it were the only one requested.

Any unrecognized option letter in a report command issues a null report (that is, a carriage return <CR>) to indicate an 'unimplemented report' to the host.

Ensure that in a terminal-to-host environment, reports that would cause errors in GIGI are not echoed to the terminal. For example, a cursor echoed to the terminal could result in a spurious vector or curve element.

The format for specifying the L command is:



Where:

(L)
Report the character set name defined for the character set currently being used to display text.

(M(*keyletter...*))
Report the contents of the macrograph(s) specified by keyletters.

Macrographs are delimited by the macrograph definition initiator and terminator just as they are defined, except that the initiator (@:) is replaced by @=. Any control characters saved with the macrograph are also returned.

(M(=))
Report the free and total macrograph storage. The report is the form:

;'*free-space/total-space*'<CR>

where *free-space* is the number of bytes still available for macrograph storage and *total-space* is the maximum number of bytes available. Remember that definitions of programmable keypad keys share space with macrograph storage.

P

Report the cursor position. The cursor position is reported as a pair of unsigned, always absolute, X-and Y-locations separated by a comma and enclosed in brackets:

[X-loc,Y-loc]

P(I [*X-min, Y-min*])

Enter interactive locator mode. GIGI halts further ReGIS processing, displays the locator cursor, and allows you to move the cursor with the cursor keys or tablet (if a tablet is attached).

When locator mode is terminated, the cursor report is sent to the host and ReGIS processing resumes.

The I option can take an optional position argument. This position argument specifies the minimum cursor movement as *X-min* and *Y-min* for each stroke of the cursor control keys.

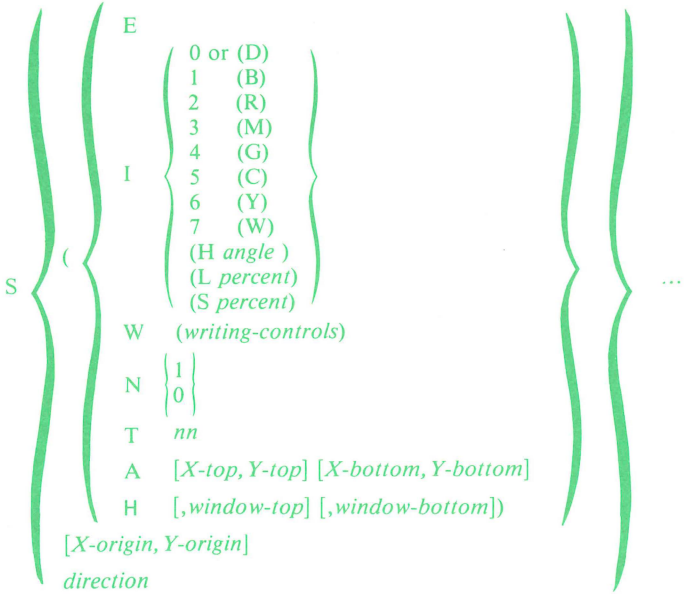
Pressing the SHIFT key and the cursor key at the same time move the cursor 10 times the minimum. If this argument is not specified, the minimum in *X-min* and *Y-min* is 1 dot.

SCREEN CONTROL COMMAND

The S (Screen) command performs the following functions:

- Erases the screen (E option).
- Specifies the screen background color (I option).
- Sets and clears reverse video for the screen (N option).
- Set temporary writing controls (W option).
- Performs a time delay before execution of ReGIS commands (T option).
- Defines the coordinate limits for the screen (A option).
- Prints a specified subset of the screen display (H option).
- Moves the screen display in a direction relative to the screen origin.

For usage information on the S command, refer to Chapter 6, *GIGI Screen Concepts*. The format for specifying the screen command is:



(E)
The screen erase option erases all images (text and drawings) on the screen.

Writing controls are not affected, except for shading, which is turned off. Also, S(E) terminates and clears curve interpolation sequences and clears all locations stored by P(B) and V(B).

The current cursor location is not changed. If the graphics origin has been moved, it is reset to [0,0].

Erasing the screen is a writing operation which, as it removes images from image memory, writes to attribute memory using the current writing control for intensity (color).

Erase has the side effect of terminating any curve interpolation sequence in progress.

(I)
The intensity option sets the background color for color monitors and the brightness for black and white monitors. Specify the screen intensity option as follows:

(I*number*)

or

(I(*letter*))

Number is a value from 0 through 7 which specifies an increasing value for shade or color. *Letter* is one of the color mnemonics listed below:

Color Number	Color Mnemonic	Color
0	(D)	Dark (black)
1	(B)	Blue
2	(R)	Red
3	(M)	Magenta
4	(G)	Green
5	(C)	Cyan
6	(Y)	Yellow
7	(W)	White

The gray scale intensity maps to the color value; therefore you can use either form of the I option to set a gray level value or a color value. For example, to set the background for a color monitor to green, you can specify either the number form, I4, or the letter form, I(G).

When GIGI is powered on or when you press SHIFT/RESET, the color is set to (I0) or(I(D)), which sets the screen to black.

Colors are described and qualified by means of hue, lightness, and saturation factors. ReGIS supports the syntax for hue, lightness, and saturation so that commands coded using these specifications can be easily transported to devices which only provide the HLS specification method. GIGI supports HLS only to the extent of the six colors and dark and white.

(N*n*)
n represents a single digit. When N is specified with a value greater than 0, for example N1, GIGI sets reverse video. N0 clears reverse video and returns to normal video. N0 is the setting at power up or SHIFT/RESET.

This option has the same effect as the RV1 set-up. When you set a negative screen, GIGI displays the video memory as if the image memory bits are inverted (0 is displayed as 1; 1 is displayed as 0). The effect of this is that the screen color attributes are used to display images and image color attributes are used to display the background.

(T *ticks*)
T directs GIGI to wait a time specified by *ticks* before executing the next ReGIS command. *Ticks* is either 60ths or 50ths of a second, depending on GIGI's power frequency setting (PF set-up). The largest number you can enter is 255, which is approximately 4 or 5 seconds.

(A [*X-top, Y-top*] [*X-bottom, Y-bottom*])
The addressing option specifies the horizontal and vertical limits for coordinates for subsequent ReGIS commands. Normally, this option should not be used with GIGI. If used, it should be specified as A[0,0][767,479].

If either or both specifiers are missing or incorrectly specified, the entire option is ignored and the display coordinates are reset to normal GIGI limits, [0,0][767,479].

(H [,*window-top*] [,*window-bottom*])
The hardcopy window option directs GIGI to print the specified subwindow of the screen on the hardcopy device.

Window-top and *window-bottom* are Y (horizontal) coordinates which specify the top and bottom limits of the screen display to be printed. GIGI ignores the X (vertical) coordinate specifications. The range is rounded up to the next 12 physical pixels; for example, when you specify (H[,10][,40]), GIGI prints the contents of the screen from row 12 through row 47.

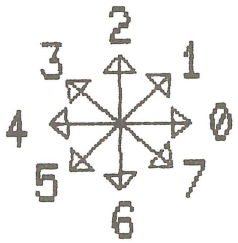
[X-origin, Y-origin]

The screen origin location option directs GIGI to move the origin of the coordinate system to another location on the screen.

GIGI moves the screen origin address, including all images currently on the screen, to the origin specified by [*X-origin, Y-origin*].

direction

The screen direction option directs GIGI to move the entire screen in the specified direction relative to the current location. You can move the screen horizontally, vertically, or diagonally in increments of eight horizontal pixels and/or twelve vertical pixels at a time. Valid entries for *direction* are:



Screen movement is relative to the current cursor position. GIGI accumulates partial offsets and moves the screen when a sufficient number of values is entered.

You can direct the screen to move horizontally, vertically, or diagonally. However, the resolution of this movement is limited to 12-pixel increments horizontally and 4-pixel increments vertically.

GIGI wraps over a vertical extent of 512 and a horizontal extent of 768. Moving the screen to the right has the effect of moving images on the screen to the left.

The writing command multiplier (M option) applies when you specify *direction*. User coordinate scaling applies if the you have specified W[*X-origin, Y-origin*].

GIGI accepts a screen addressing range which extends beyond the border of the screen. Thus, off-screen coordinates may be referenced within this range without the cursor wrapping back into the display area. This extent is subject to the dynamic range limitations of 16-bit arithmetic.

GIGI cannot be relied on to process a coordinate greater than 32767; in most cases the limit is lower than 32767.

TEXT COMMAND

The T (text) command allows you to display a text string and set the following attributes for that string:

- Size: sets the size of characters in the string.
- Direction: rotates characters or strings in the specified direction.
- Height: sets the character height.
- Italic: specifies the degree of slant for the characters in the string.
- Temporary Writing Controls: set temporary writing controls (same controls specified in the W (Write) command).
- Alphabet: specifies which of GIGI's 4 character sets to use when writing a string.
- Begin/End Pairs: save and restore text attributes.
- Multiplication Factor: explicitly specifies the multiplier for each pixel in the text cell pattern.
- Text Cell Direction: offsets the text cell in the a direction relative to the current cursor location in 1/2 cell increments.
- Character Spacing: sets writing direction and spacing between text cells.

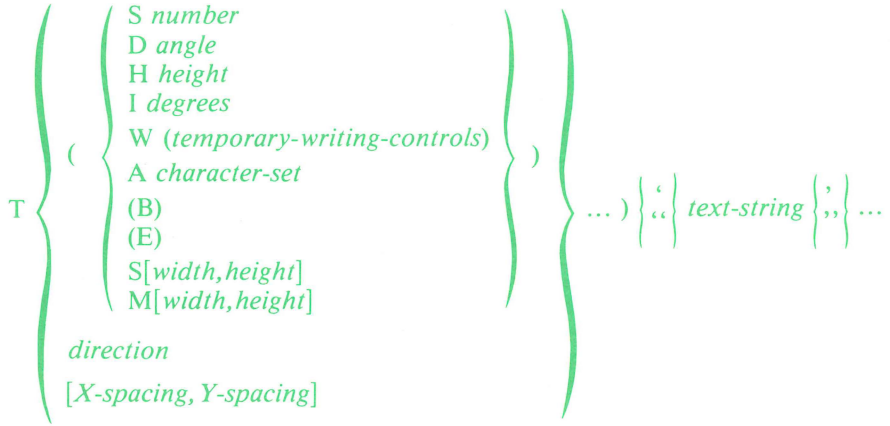
All options specified as part of the T command, except temporary write options (W), apply to all subsequent T commands until explicitly changed.

The origin for characters is the upper-left corner of the cell. Text writing does not write the pixel at the current location; rather, the first pixel written is the pixel offset by one from the current location at the start of the character. This pixel is offset in the writing direction.

Characters in a string are displayed in sequence in the direction and proportion set by character spacing, described below. Defaults for these values are described below.

For usage information on the T command, refer to the Chapter 9, *GIGI Text Concepts*.

The format for specifying the Text command is:



(S *number*)
Automatically sets the size of characters displayed in *text-string* by setting character width, height, and spacing between characters. Valid entries for the size number are 0 through 16; the default size is 1..

GIGI sets character size using the following formulas:

width = *number* X 9 pixels

height = (*number* X 1.5) X 10 pixels (rounded up)

If not set explicitly in the current T command, spacing between characters in *text-string* is based on the settings for direction and width.

You can override the height GIGI calculates by means of the H *number* option.

(D *angle*)
Sets the direction in which baseline of characters is tilted. GIGI sets direction in relationship to 0; at 0, the character baseline is horizontal (left-to-right). GIGI rounds *angle* to the nearest 45 degrees.

(H *number*)
Sets the character height to be *number* times 10 pixels.

The H *number* option provides a means for adjusting the character height after the (S *number*) option has produced the 3:2 height-to-width ratio. The height is changed to become *number* times the base character height (10 pixels).

(I *angle*)
Sets the italic slant of characters in a text string. A positive *angle* specifies cells should slant backward from the normal to the baseline by 27° or 45°, a negative *angle* specifies cells should slant forward by -27° or -45°. A zero *angle* returns the specifier to rectilinear (non-aligned).

(W(*temporary-writing-controls*))
Sets temporary writing controls, just as in the vector command. Refer to the description of the W command, later in this Chapter.

(A *alternate-character-set*)
Provides the means to specify one of GIGI's four character sets. Character set 0 is the built-in ASCII character set. Character sets 1, 2, and 3 must be defined before they can be used meaningfully. The L (load pattern) command is used to define bit patterns which are then invoked by the T command.

The pattern set selected by the (A) option is independent of the set selected in the L (load pattern) command. Changing one does not affect the other.

(B)
Saves current text attributes. (B) directs GIGI to save the current attributes in a save area.

(E)
Restores text attributes saved by (B) option. (B) and (E) options cannot be nested.

(S[*width,height*])
Explicitly sets the size (in pixel height and width) of the display frame.

Width specifies the number of horizontal pixels that define the width of the character. *Height* specifies the number of vertical pixels that define the height of the character.

GIGI uses the current size to display characters; if no Size options have been processed, GIGI displays characters in the default size, (S1).

GIGI displays as many iterations of the character pattern as will fit in a single cell.

This form of the size option does not set the spacing between characters.

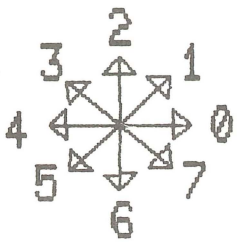
S[*width,height*] and M[*width,height*] interact and give flexibility in the display of characters and text cells.

S[*width,height*] specifies the exact width and height (in pixels) of each cell drawn; M[*width,height*] specify the number of times each dot in the text pattern is multiplied before GIGI displays the contents of the text cell. For example, these parameters can be adjusted so that only part of a character is displayed.

These forms of the Size and Multiply options are not normally used in conjunction with the (S *number*) or (H *number*) options.

(M[*width,height*])
Multiplies each dot in the text pattern as specified by *width* and *height*. Characters drawn to these specifications represent as many iterations of dots in the pattern as will fit in the display frame defined by the S[*width,height*] option.

direction
Specifies a direction relative to the current text direction, height, and width. Valid entries are 0 through 7. This value may not be parenthesized. 0 through 7 orient the upper-left corner of the text cell according to the following scheme:



Direction is relative to the current character direction. Depending on the settings for character height, width, and direction, spacing for characters is in 1/2-width row increments and 1/2-height column increments. This provides a convenient form for superscript/subscript.

[*X-spacing,Y-spacing*]
Explicitly sets the spacing GIGI uses to display characters in *text-string*. This value may not be parenthesized. This allows you to set the direction of a sequence of characters, although it does not affect the direction of the characters within the sequence.

For example, you can specify writing sequence using *X-spacing*. Thus, when you specify a negative value for *X-spacing* such as [-10,] GIGI displays the string

T[-10,]'ALLIGATOR'

as

ROTAGILLA

'*text-string*' or '*text-string*'
Specifies the characters to be displayed by the text command.

Text strings may be delimited by either a pair of double quotes (''text'') or a pair of single quotes ('text'). Two kinds of delimiters are used, but one type may be displayed by doubling it within the string.

The contents of a quoted string are displayed literally. Macrographs are neither expanded or defined, synchronization is not recognized, and any comment string within a quoted string is part of the string.

Printing characters (space through sedilla) are displayed according to the pattern stored in the currently-selected character set. All control characters except <ESC>, <TAB>, <BS>, <LF>, and <CR> are ignored. <ESC> terminates graphics mode and enters text mode.

<TAB> (horizontal TAB) produces the same character space motion as the space character, except that no pixels are written. This can be used to eliminate color contamination between adjacent words of different color. It can also produce an overstrike effect in replace mode.

<BS> (backspace) produces a reversed character space motion; in overlay writing mode it can be used to overstrike the preceding character.

<LF> (linefeed) produces a motion downward (with respect to the text cell) of length equal to the character height. (This is true except when the graphics prefix is enabled; in this case a <LF> returns GIGI to text mode).

<CR> (carriage return) returns the graphics location to the beginning location for the text string. Each time a <LF> is processed, however, that beginning location is updated by the number of pixels generated by the <LF>. The result is that multiple lines of text can be produced by a single text-string by entering <CR> <LF> pairs.

VECTOR COMMAND

The V (vector) command draws a straight line between the pixel at the current cursor location and a specified screen location. Refer to Chapter 7, *Drawing with ReGIS*, for usage information on the V command. The format for specifying the V command is:

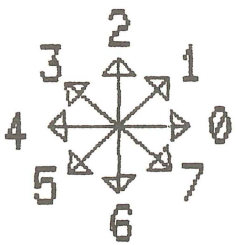
V { [ ] | [*X-loc, Y-loc*] | *direction* | (W (*temporary-writing-controls*)) | (B) | (E) } ...

[ ]
Draws a single dot at the current location. This command does not move the cursor.

[*X-loc, Y-loc*]
Specifies coordinates for a point on the screen to which GIGI draws a straight line. This command moves the current location to that point.

You may specify either relative or absolute values for *X-loc* and *Y-loc*. Refer to Chapter 6, *Screen Concepts*, for information on relative addresses.

direction
Draws a line in the direction specified by *direction* and resets the current location to that screen location. Valid entries are 0 through 7. This value may not be parenthesized. 0 through 7 draw a line in directions specified by the following scheme:



The length of the line drawn is a single point unless you set the *direction* length. Set *direction* length permanently using the Writing command multiplier (M) option; set the length temporarily using the temporary writing controls described below.

(W (*temporary-writing-controls*))
Sets temporary writing controls, just as in the W (writing) command. For information on how to specify and use writing control options, refer to the description of the W command.

Temporary-writing-controls comprise all W command options, but apply only for the extent of the current vector command. Only options specified or those implied by those specified are changed. Options return to their previous value at the end of the V command.

(B)
Directs GIGI to save the current location on a location stack; up to eight locations may be saved on this stack.

(E)
Directs GIGI to restore from the stack the most-recently entered location. In addition, V(E) draws a line back to the restored location.

The current location remains set at the last location drawn.

The drawing pattern used to draw lines is repeated cyclically during the draw operation until the end of the V command or to the end of a temporary writing control specifying a pattern. This allows a dashed line of equal marks and spaces to retain proper segment lengths while being drawn 'around corners', through curves and discontinuities.

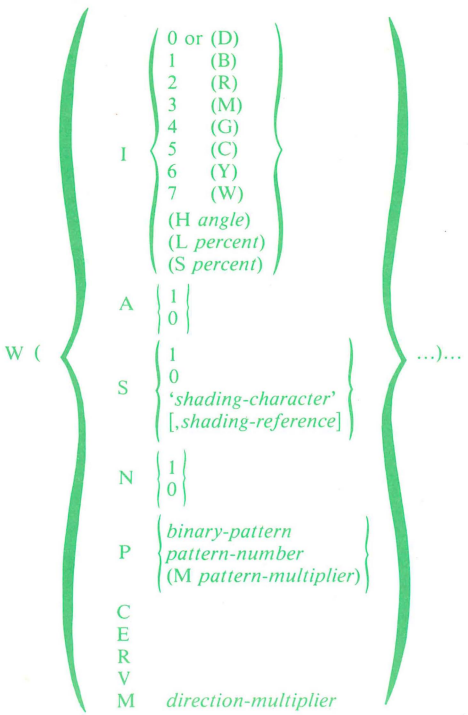
Except for V[ ], the V command never draws the pixel at the starting location. However, the endpoint of each vector segment is drawn. This allows you to draw a sequence of vectors using complement writing.

WRITING COMMAND

The W (Writing) command performs the following functions:

- Sets the color GIGI uses for drawing pictures and displaying text (the I option).
- Directs GIGI to alternate the display (blink) of subsequently-specified screen images (the A option).
- Enables area shading; specifies the character with which to shade the area (the S option).
- Enables or disables negative writing (the N option).
- Specifies the writing pattern used to draw lines and curves (the P option). The P option also allows you to specify a pattern multiplication factor (P(M)).
- Specifies logical operations to perform using the writing pattern:
  C - Complement R - Replace
  E - Erase V - Overlay
- Specifies the length in pixels of offset direction operations (M option).

Refer to Chapter 8, *GIGI Writing Concepts*, for usage information and examples of the W command. The format for specifying the W command is:



(I)
The intensity option sets the writing color monitors or brightness for black and white monitors. Specify the writing intensity option as follows:

(I*number*)
or
(I(*letter*))

Number is a value from 0 through 7 which specifies an increasing value for shade or color. *Letter* specifies one of the color mnemonics listed below.

Color Number	Color Mnemonic	Color
0	(D)	Dark (black)
1	(B)	Blue
2	(R)	Red
3	(M)	Magenta
4	(G)	Green
5	(C)	Cyan
6	(Y)	Yellow
7	(W)	White

The gray scale intensity maps to the color value; therefore you can use either form of the I option to set a gray level value or a color value. For example, to set the writing color for a color monitor to green, you can specify either the number form, I4, or the letter form, I(G).

When GIGI is powered on or when you press SHIFT/RESET, GIGI sets the writing color to (I7) or (I(W)) (white) and writing mode to replace writing.

Colors may be described and qualified by means of hue, lightness, and saturation factors. ReGIS supports the syntax for hue, lightness, and saturation so that commands coded using these specifications can be easily transported to devices which only provide the HLS specification method. GIGI supports HLS only to the extent of the six colors and dark and white.

The I option with no parameters directs GIGI to write without changing the current attributes. Pixels drawn with this setting have whatever color and blink attributes set for that location on the screen. This is 'white, no blink' if nothing has been drawn in that attribute block. This setting is cancelled whenever an I with a parameter is specified and whenever an A option is specified.

(A*n*)
Enable the alternate (or 'blink') attribute. *n* is either 0 or 1. A0 turns off the blink attribute; A1 turns on the blink attribute. A0 is the default.

(S*n*)
The shading option directs GIGI to shade from the current shading reference during subsequent write operations. *n* is either 1 or 0. (S1) enables shading; (S0) disables shading. The shading option enables or disables area shading. When you specify a nonzero digit, the current drawing pattern and pattern multiplier (see the P option, below) is used to fill areas subsequently drawn.

To shade with a writing pattern, specify the pattern before turning shading on. Writing patterns are described below.

(S'*shade-character*')
Directs GIGI to use the character specified by shade-character when shading.

The shading pattern for the W command operates on the character set currently specified by the T command. GIGI uses the pattern set in the character cell corresponding to the ASCII character you enter as shade-character. You can set the current character set by means of the T (text) command A (alphabet) option. The Text command M option multipliers affect the appearance of the shading pattern, as does the T command I (italic) option.

(S[,*shading-reference*])
Defines the horizontal reference line for area shading. ReGIS shades an area between this horizontal reference line and the line currently being drawn. The reference line may be specified either explicitly with this option or, by default, is the horizontal line on which the cursor is positioned when the S option is encountered.

Remember that the shading option is not an automatic fill function for arbitrary areas. Filling of objects depends on correct placement of the shading reference line or lines as determined by the shape of the object.

(N*n*)
The negative writing option directs GIGI to set or clear negative (reverse) writing. *n* may be either 0 or 1. N1 sets reverse writing; N0 clears reverse writing. N0 is the default.

Negative (reverse) writing directs GIGI to invert writing pattern bits before they are entered into display memory. For example, if the writing pattern is 11000000, N1 interprets the pattern as 00111111.

Negative writing also reverses the effect of erase writing; N1 causes erase writing to force image bits to 1, whereas N0 forces image bits to 0.

(P*binary-pattern*)
Specifies a 2- to 8-bit pattern you can set with 1's and 0's. When GIGI draws on the screen, bits that are set to 1 are drawn, bits that are set to 0 appear as gaps in the line.

For example, P1100 draws lines dashed with equal gap and mark spacing; P11100111 draws dashed lines with marks three times as long as gaps.

The maximum length of the pattern is 8 bits. Patterns are stored as 8-bit integers, regardless of the specified length. Lengths of 2, 4, and 8 bits are repeated exactly. Lengths of 3, 5, 6, and 7 bits are repeated twice, once, once, and once, respectively in each 8 bits, with the first 2 bits, 3 bits, 2 bits, and 1 bit, respectively, filling out the remainder of each 8-bit length.

P101 draws a pattern 1011011010110110. Note that pattern does not repeat in length 3.

(P*pattern-number*)
Selects a predefined pattern from the list below. Predefined linear pattern specifiers 0 through 9 are defined by the following binary patterns:

0 P00000000 Blank pattern (no dots set)
1 P11111111 Solid line
2 P11110000 Dash pattern
3 P11100100 Dash dot pattern
4 P10101010 Dot pattern
5 P11101010 Dash dot dot pattern
6 through 9 Unspecified

Pattern 1 is the setting at power up and SHIFT/RESET.

Patterns 6 through 9 produce a visible effect when you specify them; compatibility with other ReGIS terminals is not guaranteed, however.

The number of pixels represented by a bit in the pattern is equal to the value of the *pattern-multiplier*, described below.

(M *pattern-multiplier*)
Defines a multiplier for each bit in the current writing pattern.

Use of the *pattern-multiplier* increases the variety of the patterns shown. W(P4(M1)), W(P4(M2)), and W(P4(M3)) produce three visibly different drawing patterns.

(V)
Specifies overlay writing. Overlay writing directs GIGI to change a pixel bit to 1 if the writing pattern bit is 1 and not change the pixel bit if the writing pattern bit is 0. Thus if a pixel bit is 1, it stays 1; if a bit is 0, it is changed to 1 only if the writing pattern bit is 1. Overlay writing is GIGI's initial setting when powered on.

Refer to Chapter 8, *Writing Concepts*, for an example of overlay writing.

(R)
Specifies replace writing. Replace writing forces the pixel bit to be the same as the pattern bit (either 0 or 1) regardless of the current value of the pixel bit.

Refer to Chapter 8, *Writing Concepts*, for an example of replace writing.

(E)
Specifies erase writing. In erase writing, the writing pattern is not used. Normally, erase writing forces each bit written to be 0, unless W(N1) (negative writing) is set. If negative writing is set, each bit written is forced to 1.

Refer to Chapter 8, *Writing Concepts*, for an example of erase writing.

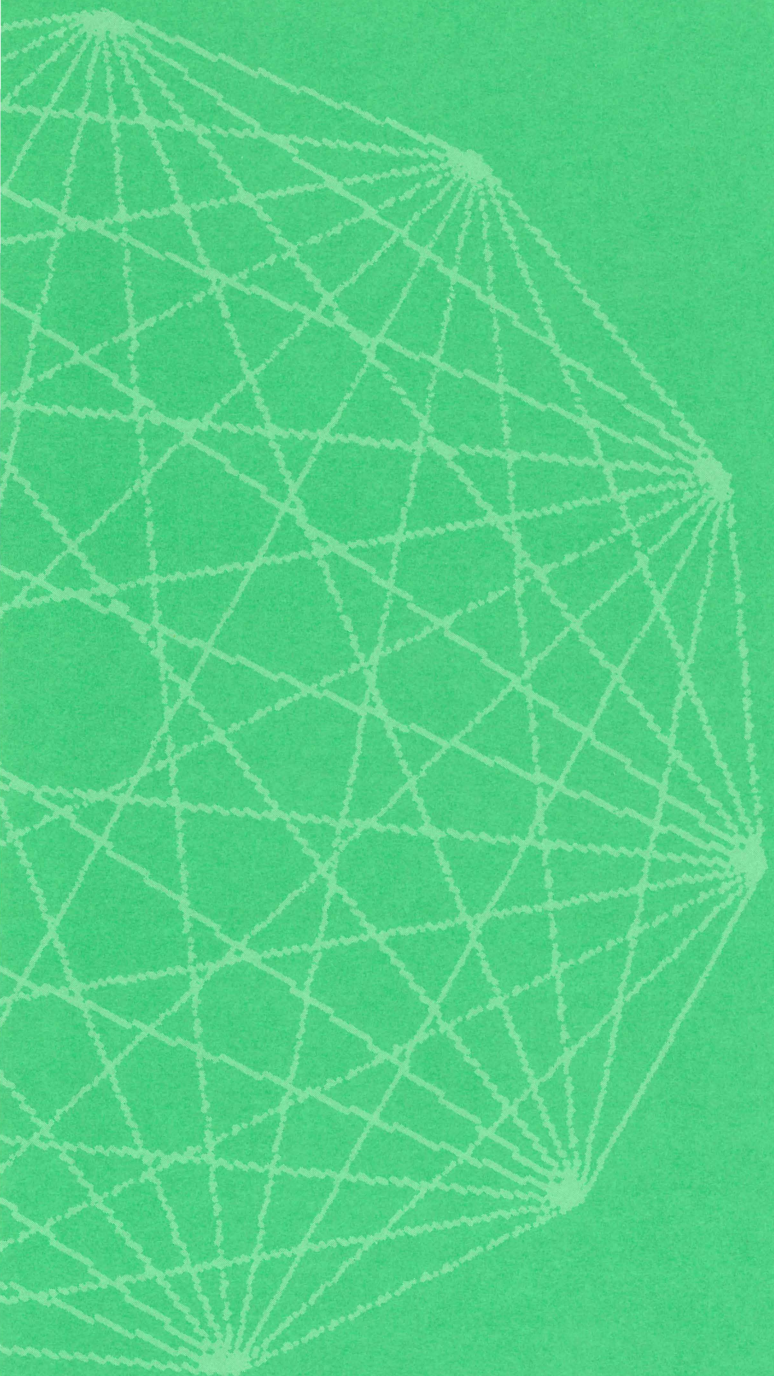
(C)
Specifies complement writing. In complement writing, if the pattern bit is 0, GIGI does not change the pixel bit being written. But if the pattern bit is 1, GIGI changes the corresponding pixel bit to 1 if it is 0 or to 0 if it is 1.

Refer to Chapter 8, *Writing Concepts*, for an example of complement writing.

(M *direction-multiplier*)
The direction multiplier option is used with the C, V, P, and S commands to replicate the number of pixels drawn or moved by means of *direction* specifications in those commands. For example, GIGI draws 30 pixels for each value entered in the following V command:

```
W(M30)
V642446064600206
```

3



Programming GIGI

This part of the manual contains:

Chapter 13 contains examples of programming graphics sequences using GIGI BASIC and PASCAL.

Chapter 14 provides information on how to program the programmable keypad.

Chapter 15 provides information on locator mode.

Chapter 16 provides information on escape sequences.

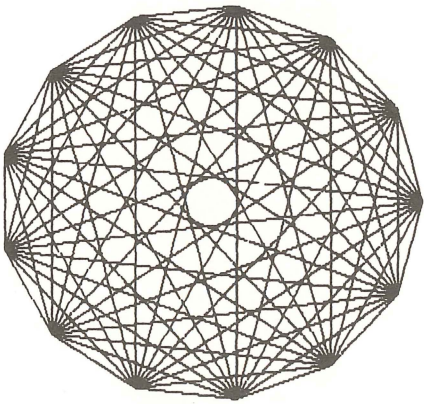
13 PROGRAMMING PICTURES

This chapter contains examples of ReGIS programs for drawing:

- Rosette
- Spiral
- Shading example in BASIC
- Bubble: Animation Example Using Macrographs
- Towers of Hanoi: Animation Example in PASCAL

ROSETTE

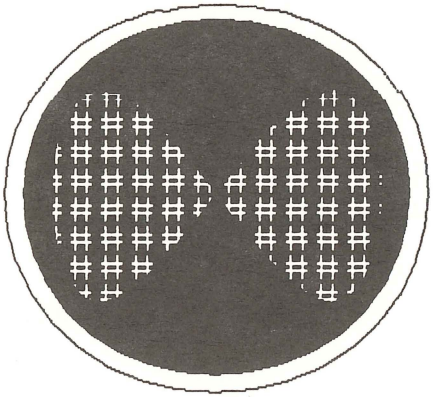
Use the code in this example to draw a 13-point rosette. This program is useful in demonstrating the speed of ReGIS code executing from macrograph storage versus the speed of code being generated by the host.



```
S(E)                        ;'erase screen'
W(R,I(G),P1,N0,A0,S0)       ;'replace mode, green, solid lines, turn off: negative writing,
                             blink, shading'
T(A0,D0,I0,S2)              ;'prepare to write title with size 2 characters'
P[20,20]T" 13 POINTS"       ;'write title'
@.                          ;'clear any previous macrographs'
@:BV[ 620 , 240 ]@;         ;'define a macrograph to draw a vector to each vertex'
@:CV[ 595 , 342 ]@;
@:DV[ 525 , 421 ]@;
@:EV[ 427 , 458 ]@;
@:FV[ 322 , 446 ]@;
@:GV[ 235 , 386 ]@;
@:HV[ 186 , 293 ]@;
@:IV[ 186 , 187 ]@;
@:JV[ 235 ,  94 ]@;
@:KV[ 322 ,  34 ]@;
@:LV[ 427 ,  22 ]@;
@:MV[ 525 ,  59 ]@;
@:NV[ 595 , 138 ]@;
                            ;'A is the big macrograph that draws all connections'
@:AP[ 620,  240]
@C@D@E@F@G@H@I@J@K@L@M@N@B
@D@F@H@J@9@N@C@E@G@I@K@M@B
@E@H@K@N@D@G@J@M@C@F@I@L@B
@F@J@N@E@I@M@D@H@L@C@G@K@B
@G@L@D@I@N@F@K@C@H@M@E@J@B
@H@N@G@M@F@L@E@K@D@J@C@I@B
@;
@A                          ;'execute everything'
```

SHADING AND CIRCLES EXAMPLE

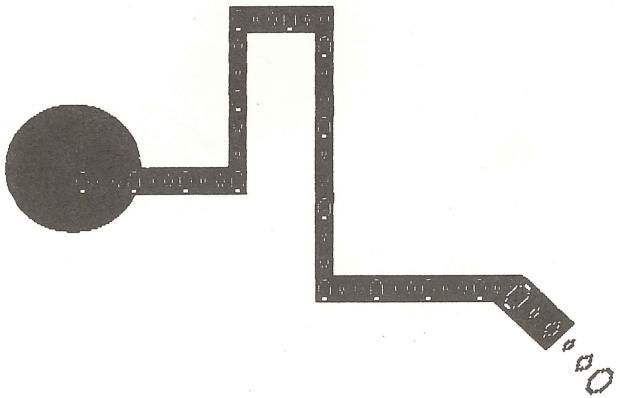
This code demonstrates how to use various writing options.



```
S(E)                    ;'erase the screen'
W(R,I7,P1,N0,A0,S0)     ;'replace writing mode, white, solid pattern, turn off
                          negative writing, blink, and shading'
P[380,260]              ;'position at center of circles'
W(I(G))                 ;'write with green'
C[+220]                 ;'circle of radius 220'
W(I(R),S1)              ;'write with red, with shading enabled'
C[+200]                 ;'circle of radius 200'
T(M[4,4])               ;'set both character pattern multipliers to 4'
W(N1,S"#")              ;'turn on negative writing, enable shading with "#" pattern'
P[-120,-120]            ;'change position to upper left of center of circles'
C(B)                    ;'begin a closed-curve interpolation sequence'
                        ;'first point of curve is current position (up/left) second
                          point is below and right of center, third is above and right of
                          center, fourth is below and left of center'
C[+240,+240]
[,-240]
[-240,+240]
C(E)                    ;'end the sequence, close the figure'
```

BUBBLE ANIMATION

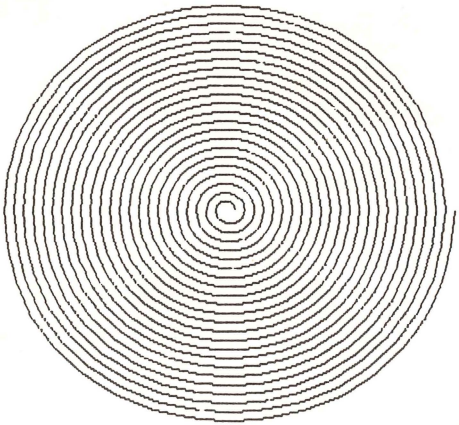
This example demonstrates how to animate a picture using macrographs and alternate character sets.



```
;('bubble alphabet')
L(a1)
L'A'00000008141408;
L'B'0000081422221408;
L'C'001C2241414141221C;

L(a2)
L'A'001C2241414141221C;
L'B'00000008141408;
L'C'0000081422221408;

L(a3)
L'A'0000081422221408;
L'B'001C2241414141221C;
L'C'00000008141408;

;'pipeline'
@:P W(N1)p(b)t[+18,0]"ABCABCABC"[0,-30]"ABCABC" @Q p(e)@;
@:Q t[+18,0]"ABCAB"[0,+30]"CABCABCABC"[+18,0]"ABCABCABCABC" @N
@
;
@:N T(bD-45s2)[+16,+16]"ABC"(W(N0))"ABC"(e) @;

;'flashing'
@:X t(a1)@Pt(a2)@Pt(a3)@P @;
S(ei0)w(i7n0cs0)p[200,200]
t(l0S2)w(s1ri(b)a0);'italics off; size two; shading on; color blue; flash of f'
c[+70]
@X@X@X@X@X@x@x@x@x@x@x@x@x@x@x
```

BASIC PROGRAM FOR DRAWING SPIRAL

This GIGI BASIC program demonstrates the use of GIGI BASIC to create graphics.

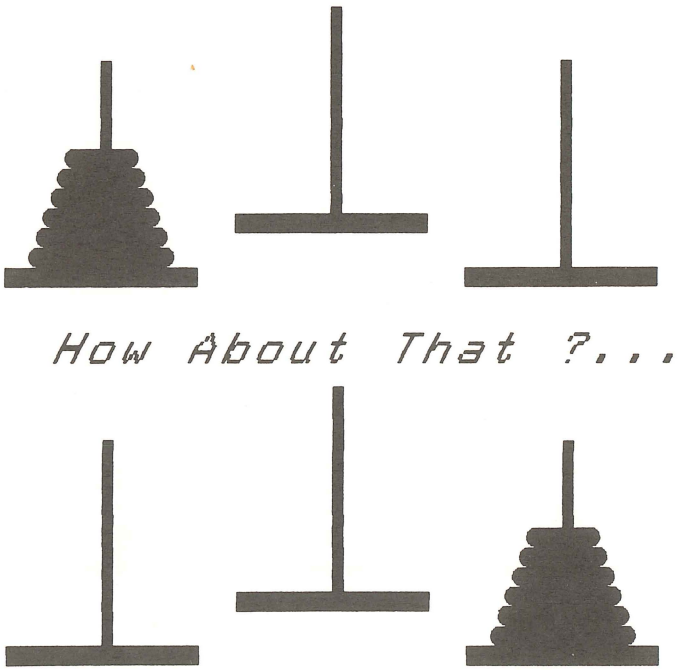


```
10    REM DISPLAY A MULTI-COLOR SPIRAL PATTERN
20    REM REQUIRES SET-UP "GP1" (GRAPHICS PREFIX)
30    REM FIRST CLEAR SCREEN, SET UP WRITING CONTROLS, POSITION AT CENTER
40    PRINT "!S(EI0)w(p1,r,n0,a0)P[350,240]"
50    REM APPROACH IS TO DRAW 23 SEMI-CIRCLES, EACH STARTING WHERE THE PREVIOUS
60    REM SEMI-CIRCLE ENDED; COLOR IS A FUNCTION OF THE ITERATION COUNT
70    FOR I=1 TO 23
80    PRINT "!W(I"; INT(1+I/24 7); ")"
90    PRINT "!C(A180C)[-"; I*10; "]"
100   PRINT "!C(A180C)[+"; I*10+5; "]"
110   NEXT I
120   REM HOME TEXT CURSOR TO PREVENT SCROLLING WHEN DONE
130   PRINT CHR$(155)+"[H"
999   END
```

TOWERS OF HANOI: ANIMATION USING PASCAL

This program shows how the ReGIS graphics language is easily adapted for use in drawing pictures with higher level languages.

The first screen below is first frame of the animation; the second screen is the last frame of the animation.




```
program gigihanoi(input,output);
(*towers of hanoi*)
(*problem is to move n discs from pole a to pole c such that
     (1)initially discs on a in increasing size order.
     (2)one disc moved at a time
     (3)a disc is never on a smaller disc
     (4)the third pole(b) may be used in the move.
the problem is solved recursively:
       move n-1 from a to b
  move the nth disc from a to c(this is printed by the program)
  move n-1 from b to c*)

const chsa = ' p[255,160]@';
      chsb = ' p[385,100]@';
      chsc = ' p[495,160]@';
var P1,P2,P3,nud,n:integer;
    mid : packed array [1..12] of char;
```

```
procedure drawdisk(disk,height:integer);
begin
  height := -22*height+10;
  if height>-100 then writeln(' p[,',height:3,']') else
  writeln(' p[,',height:4,']');  (*position dot at height*)
  disk := disk+64;
  writeln(' @',chr(disk));  (*draw disk by calling the macrograph*)
end (*drawdisk*);

procedure drawmid(tdsk:integer);
  begin
    tdsk := tdsk+64;
    writeln(mid:12,chr(tdsk));
  end (* drawmid *);

procedure initialize(n:integer);
var alpha:char; dw,dw1,i,wc,col:integer;
begin
  writeln(' ',chr(27),'Pp');  (*<gon>*)
  writeln(' S(E,I1)');
  writeln(' P[345,130]');
  writeln(' W(I(R),-A,-N,P1)');
  writeln(' T(A0,s4h4,I0)''THE''P[80,210]');
  writeln(' T(I0,s5h5)''TOWERS OF HANOI''P[200,330]');
  writeln(' W(I(Y))T(I0,s2h2)''(one moment, please)''');
  writeln(' W[+1](P1,R,-S,-N,I6)S(I1)');
  alpha:=chr(64);
  writeln(' @.@:Pw(+s,-n,i6)p[,-20]v[+10]@;');
  for i:=1 to 10 do
    begin
      DW:=55+15*(I-1);
      ALPHA:=SUCC(ALPHA);
      wc:=ord(alpha)-64;
      case wc of
        1,4,7,10 : col := 2;
        2,5,8 : col := 7;
        otherwise col := 4
      end;
      dw1:=round(dw/2);
      writeln('@:',alpha,'w(+s1,-n0,i',col,')p[+',dw1:3,'](b)c(A-180)[,-10]
          p(e)[,+10]w(+s1)V[,-20][-',DW:3,']');
      writeln(' P[,+10]w(+s1)C(A-180)[,+10]@;');
    end (* build disk macrographs *);
writeln(' @:Qw(+s,-n)p[,-250]v[+10]p[-110,+230]v[+200]@;');
writeln(' s(e)');
writeln(' p[135,450]@Qp[+140,-40]@Qp[+140,+80]@Q');
(* we build the poles with bases *)
P1:=0;
for i:=n downto 1 do
  begin
    writeln(' p[135,430]');
    P1:=P1+1;
    drawdisk(i,P1);
  end;
end (*initialize*);
```

```
procedure fixpole(height:integer);
begin
  height:=-22*height+20;
  writeln(' p[,',height:3,']');
  writeln(' @P');
end (*fixpole*);

procedure movedisk (d:integer;x,y:char);
begin
  writeln(' w(r)');  (* write replace mode*)
  if x = 'a'
    then
      begin
        if y = 'b' then mid:=chsa else mid := chsb;
        drawmid(d);
        writeln(' p[135,430]w(e)');
        drawdisk(d,P1);
        writeln(' w(r)'); (*set write*)
        writeln(' P[135,430]');
        fixpole(P1);
        P1:= P1-1;
      end;
  if x = 'b'
    then
      begin
        if y = 'a' then mid:=chsa else mid:=chsc;
        drawmid(d);
        writeln (' p[375,370]w(e)');
        drawdisk(d,P2);
        writeln(' w(r)'); (*set write*)
        writeln(' p[375,370]');
        fixpole(P2);
        P2:=P2-1;
      end;
  if x = 'c'
    then
      begin
        if y = 'a' then mid:=chsb else mid:=chsc;
        drawmid(d);
        writeln (' p[615,430]w(e)');
        drawdisk(d,P3);
        writeln(' w(r)'); (*set write*)
        writeln(' p[615,430]');
        fixpole(p3);
        P3:=P3-1;
      end;
  if y = 'a'
    then
      begin
        P1:=P1+1;
        writeln (' p[135,430]');
        drawdisk(d,P1);
        write(' w(e)'); (* erase middraw*)
        if x = 'b' then mid:=chsa else mid:=chsb;
        drawmid(d);
      end;
```

```
  if y = 'b'
    then
      begin
        P2:=P2+1;
        writeln(' p[375,370]');
        drawdisk(d,P2);
        write(' w(e)'); (* erase middraw *)
        if x = 'a' then mid := chsa else mid := chsc;
        drawmid(d);
      end;
  if y = 'c'
    then
      begin
        P3:=P3+1;
        writeln(' p[615,430]');
        drawdisk(d,P3);
        write(' w(e)'); (* erase middraw *)
        if x = 'a' then mid := chsb else mid := chsc;
        drawmid(d);
      end;
end  (*movedisk*);

procedure hanoi(n:integer;a,c,b:char);
begin
  if n>0 then
    begin
    hanoi(n-1,a,b,c);
    movedisk (n,a,c);
    hanoi(n-1,b,c,a)
    end
end(*hanoi*);

begin(*main*)
writeln(' Number of Disks?');
readln (nud);
n:=nud;
initialize(n);
   P2:=0;
   P3:=0;
   hanoi(n,'a','c','b');
   writeln (' p[100,60]w(i6,r)t(A0,s3h3,d0,i7)[+35,+0]''How About That ?...''');
WRITELN (' ',CHR(27),'\');
end.
```

14 PROGRAMMING THE AUXILIARY KEYPAD

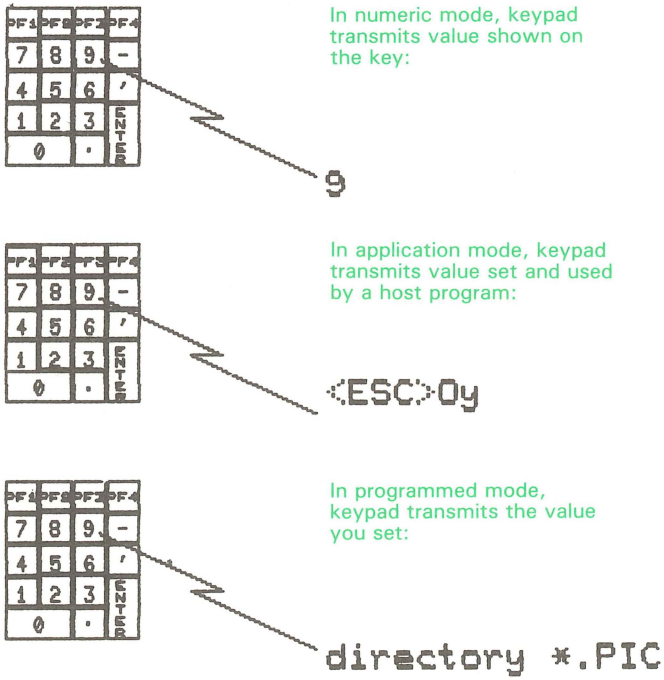
GIGI's auxiliary keypad can operate in three modes, depending on the set-ups you select. (Figure 14-2, below, shows the relationship of set-up values and terminal operating modes.) The auxiliary keypad keys can be set to:

- Numeric mode
- Application mode
- Programmable mode

In numeric mode, the auxiliary keypad keys transmit the numbers or characters shown on the keys. In application mode, the auxiliary keypad keys transmit predefined ANSI or VT52 escape sequences listed in Table 14-1, below. In programmable mode, the auxiliary keypad keys transmit characters you define using the Device Control String described in Chapter 16, *Programming With Escape Sequences*.

Figure 14-1 illustrates these three modes of operation; the codes generated depend on whether the terminal mode is VT52 or ANSI, as listed in Table 14-1, below. In each of the three keypads shown, the 9 key is set to a different value, one value for each mode. In numeric mode, the 9 key generates the character 9, just what you see on the keypad. In application mode, the 9 key is set to generate either a VT52 or ANSI escape sequence meaningful to a host program. In programmed mode, the 9 key generates an operating system command set by a user.

Figure 14-1.
Examples of Values Transmitted by Auxiliary Keypad in Three Modes



The following BASIC program illustrates a method for interactively setting the keypad keys.

```
1       REM this program runs on BASIC-PLUS or VAX-11 BASIC;
3       REM it can easily be modified to run in GIGI BASIC
5       EXTEND
8       REM define constants for ANSI control strings
10      ESC$ = CHR$(155)
13      SETDCS$ = ESC$ + "Pr" \ ST$ = ESC$ + "\"
30      INPUT "Do you want to clear all current programmed keys"; L$
40      IF LEFT(CVT$$(L$,34%),1) <> "Y" THEN 80
45      REM clear each of the keys by setting them to a zero-length definition
50      PRINT "OK -- clearing."
55      FOR I = 0 TO 21
60      PRINT ESC$ + "P" + NUM1$(I) + "s" + ST$;chr$(13); \ REM do a CR, no LF
70      NEXT I
80      REM the following array allows a typed-in key name
85      REM to be translated to a key number
90      DIM S$(30)
100     S$(0%) = '"0"'
101     S$(1) = '"1"'
102     S$(2) = '"2"'
103     S$(3) = '"3"'
104     S$(4) = '"4"'
105     S$(5) = '"5"'
106     S$(6) = '"6"'
107     S$(7) = '"7"'
108     S$(8) = '"8"'
109     S$(9) = '"9"'
110     S$(10) = 'Enter'
111     S$(11) = '"-"'
112     S$(12) = '",","'
113     S$(13) = '".","'
114     S$(14) = 'PF1'
115     S$(15) = 'PF2'
116     S$(16) = 'PF3'
117     S$(17) = 'PF4'
118     S$(18) = 'Up'
119     S$(19) = 'Down'
120     S$(20) = 'Left'
121     S$(21) = 'Right'
200     REM programmed keypad is turned off for the next input only
210     REM so user can use numeric keys as themselves
220     PRINT SETDCS$ + "TM1PK0CK0KP0" + ST$;
223     PRINT "Name of the keypad key:" \INPUT LINE K$ \ K$ = CVT$$(K$,2 + 4 + 32)
226     PRINT SETDCS$ + "PK1" + ST$;
230     IF LEN(K$) < 1 THEN 1000
250     FOR K = 0 TO 21 \ REM find the typed in key name in the list
260     IF INSTR(1,CVT$$(S$(K),32),K$) > 0 GOTO 320
270     NEXT K
280     GOTO 220
300     REM read the string associated with the key and program it
320     PRINT 'String for ' + S$(K) + ' key';
330     INPUT LINE T$
```

continued next page

```
340    PRINT ESC$+'P'+NUM1$(K)+'s'+CVT$$(T$,4%)+ST$;
350    GOTO 220
999    REM give the user a chance to try out the keys
1000   PRINT "Now try out the keys, exit with CTRL-C"
1010   INPUT LINE L$   \ GOTO 1010
9999   END
```

You control ANSI and VT52 escape sequences and their relationship to the auxiliary keypad using GIGI's set-ups. Figure 14.1 illustrates the relationships among GIGI's terminal mode set-ups.

Figure 14-2. GIGI Set-ups for Controlling the Programmable Keypad

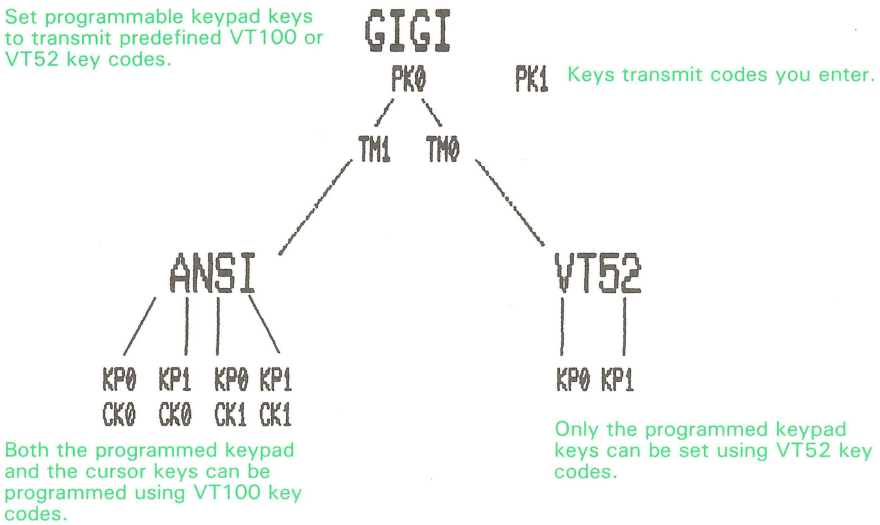


Table 14-1 lists the code GIGI transmits in numeric mode and in application mode. These codes can be transmitted either to the host when GIGI is online (set-up LL1) or to the terminal when GIGI is in local mode (set-up LL0).

Table 14-1. Codes Generated by Auxiliary Keypad

Key	Numeric Mode	Application Mode (ANSI)	(VT52)
0	0	<ESC>Op	<ESC>?p
1	1	<ESC>Oq	<ESC>?q
2	2	<ESC>Or	<ESC>?r
3	3	<ESC>Os	<ESC>?s
4	4	<ESC>Ot	<ESC>?t
5	5	<ESC>Ou	<ESC>?u
6	6	<ESC>Ov	<ESC>?v
7	7	<ESC>Ow	<ESC>?w
8	8	<ESC>Ox	<ESC>?x
9	9	<ESC>Oy	<ESC>?y
,	,	<ESC>Om	<ESC>?m
-	-	<ESC>Ol	<ESC>?l
.	.	<ESC>On	<ESC>?n
ENTER	Same as RETURN	<ESC>OM	<ESC>?M
PF1	Same as in application mode	<ESC>OP	<ESC>P
PF2		<ESC>OQ	<ESC>Q
PF3		<ESC>OR	<ESC>R
PF4		<ESC>OS	<ESC>S

PROGRAMMABLE KEYPAD SET-UPS

The PK set-up determines whether the programmable keypad transmits keycodes you set by means of the Device Control String or keycodes defined in the terminal. PK1 directs GIGI to transmit codes you set; PK0 directs GIGI to transmit codes defined by the terminal.

If PK0 is set, the Terminal Mode set-up takes effect. TM1 directs GIGI to transmit ANSI escape sequences; TM0 directs GIGI to transmit VT52 escape sequences. These escape sequences can be interpreted by host programs to perform such functions as line or word erase directly on the monitor screen. The TM set-up affects both the KP (auxiliary keypad) set-up and the CK (cursor keys) set-up. If TM0 (VT52 escape sequences) is set, only the KP set-up is used; the CK set-up has no effect. If TM1 (ANSI escape sequences) is set, you can use KP to program all the keys on the auxiliary keypad and the CK setup to program the cursor keys.

15 LOCATOR MODE

Locator mode is a graphics mode in which GIGI allows you to control the location GIGI is referencing (that is, the current location) by means of:

- A graphic cursor
- Arrow keys
- SHIFT key

Also, you can place GIGI in locator mode by entering the ReGIS R (Report) command, as described below.

Use locator mode to report screen locations to a host program.

ENTERING LOCATOR MODE

GIGI enters locator mode via either the ReGIS R (Report) command or when you press SHIFT/PF2.

The R (Report) Command

You can direct GIGI to enter locator mode by transmitting the ReGIS R (report) command in the form:

R(P(I))

P
directs GIGI to transmit the current position to the host.

I
directs GIGI to enter locator mode. This allows you to enter locator mode interactively from the host.

LOCATOR MODE OPERATION

When the terminal is in locator mode, the locator cursor (a large cross-hair) appears on the display.

When you press the arrow keys, the cursor moves a single increment at a time. Pressing the SHIFT key and the arrow key at the same time moves the cursor in 10-dot increments. Attempting to move the cursor beyond the edge of the display leaves the cursor at the edge of the display.

When you press a code-transmitting key, for example the character V, GIGI exits from locator mode and reports that character followed by the current location to the host:

V[X-loc,Y-loc]

This sequence is useful for reporting a vector command back to a host program, as described in the section *Reporting Locations to Host Programs*, below.

When you press the <CR> (carriage return) key, GIGI exits locator mode and reports only the current location to the host.

When you press the <DEL> (delete) key, GIGI leaves locator mode and does not report a location to the host.

Using a Graphics Tablet in Locator Mode

If SETUP mode TL1 is set and a digitizer tablet is connected to the hard-copy port (RS232, 9600 baud, ASCII, stream mode), the tablet cursor position overrides the cursor position keys. All other keys function as usual to terminate locator mode, as will the tablet cursor buttons or tip switch.

Reporting Locations to Host Programs

As described above, when you press a code-transmitting key such as the V key, GIGI reports that character along with the current location. This location can be read by a host program, as shown in the example BASIC program below:

```
10      ESC$ = CHR$(155)
15      GON$ = ESC$ + "Pp"
20      GOFF$ = ESC$ + "\ "
30      PRINT GON$;"W(M10,P1,I7,V)S(E)V[200,200]"
40      PRINT "R(P)"                                    !send the location
50      INPUT LINE A$
52      PRINT "R(P(I))"                                 !set interactive
55      INPUT LINE B$
57      PRINT GOFF$; \ IF LEFT(B$,1) <> "[" THEN STOP
60      PRINT ESC$;"[H";A$;B$;GON$
65      PRINT "P";A$;"V";B$
70      GOTO 40
9999    END
```

16 PROGRAMMING WITH ESCAPE SEQUENCES

An *escape sequence* is a sequence of characters, preceded by the <ESC> character, which have a specific meaning to a given device. Using escape sequences, you can program GIGI to perform both graphics mode functions and text mode functions.

In text mode, escape sequences direct GIGI to perform such functions as move the cursor or erase a line of text and set terminal operating characteristics such as keyclick and reverse video.

In graphics mode, GIGI interprets a special escape sequence and draws images on the monitor screen. Information on GIGI graphics mode is in Part 2, *GIGI Graphics*.

GIGI is designed to intercept escape sequences either directly from the keyboard (local mode) or from a host computer system (on-line mode).

There are two general types of escape sequences: simple sequences that contain no variable fields within the sequence and Device Control Strings, which are escape sequences that contain variable fields.

ESCAPE SEQUENCES

Specify simple escape sequences by entering:

<ESC>*sequence*

where *sequence* may be any of the strings of characters described in the following sections.

DEVICE CONTROL STRINGS

Device Control Strings (DCS) are escape sequences which contain variable fields. The variable fields enable them to be easily programmed to fit many applications. For example, you can index at a certain column number by setting that column number by means of a variable in a DCS.

There are four GIGI-specific Device Control Strings. The paragraphs bel describe them and the functions they perform.

GIGI-Specific Device Control Strings

GIGI supports four device control strings that perform the following fur tions using GIGI's features:

- Enter and exit graphics mode.
- Transmit screen images to the graphics printer.
- Set GIGI set-up parameters.
- Program keys on the auxiliary keypad.

The general form for specifying GIGI-specific Device Control Strings is

$$<\text{ESC}>\text{P}\left\{\begin{matrix} p \\ q \\ r \end{matrix}\right\} \textit{control-string};\ldots<\text{ESC}>\ \backslash$$

where:

<ESC>P
is the device control string prefix and must be specified at the beginning of any device control string.

The keyletters *p*, *q*, *r*, and specify the type of control string being transmitted to GIGI and determine how GIGI interprets *control-string*. All of these characters are lowercase characters.

p
Specifies that *control-string* consists of ReGIS commands. GIGI directs the entire string to the ReGIS interpreter for processing. Refer to Part II, *GIGI GRAPHICS*, for a description of how to use the Device Control String with GIGI graphics commands.

q
Specifies that *control-string* is encoded raster data to be transmitted to the LA34-RO printer.

r

Specifies that *control-string* is a set-up parameter; GIGI reads and interprets the specified set-up. Refer to the section *Programming GIGI Set-ups*, below, for a description of how to use the device control string to set set-up parameters from a program.

The device control string for programming the auxiliary keypad in programmable keypad mode (PK1 set-up) is:

<ESC>P*key-value s string*<ESC> \

<ESC>P
is the prefix, as described above.

key-value
associates *string* with a specific key on the programmable keypad:

Value	Key	Value	Key
0	0	12	,
1	1	13	.
2	2	14	PF1
3	3	15	PF2
4	4	16	PF3
5	5	17	PF4
6	6	18	Up
7	7	19	Down
8	8	20	Left
9	9	21	Right
10	ENTER		
11	-		

s

Specifies that the *string* is a string to be transmitted to GIGI by a key on the programmable keypad.

If no *string* follows, the current definition is deleted and the key transmits the *value* printed on the keycap.

Set-up PK1 (programmed keypad mode) must be enabled to activate these definitions.

TEXT MODE ANSI CONTROL SEQUENCES

Escape sequences are used to control cursor movement, erase portions of the screen, set terminal characteristics such as screen and writing color, and to program the LED's (indicator lights) on the keyboard. The following tables list the ANSI escape sequences that perform these functions.

ANSI Cursor Movement Escape Sequences

Cursor Up <ESC>[*value*A	Move current location upward without altering the column position. *value* specifies number of lines moved; default is 1. A *value* of 0 or 1 moves the current location one line upward. A *value* of n moves the current location n lines upward.
Cursor Down <ESC>[*value*B	Move current location downward without altering the column position. *Value* specifies number of lines moved; default is 1. If *value* is 0 or 1, the current location is moved one line downward. If *value* is n, the current location is moved n lines downward.
Cursor Forward <ESC>[*value*D	Move the current location to the right. *Value* specifies the distance to be moved; the default is 1. A *value* of 0 or 1 moves the current location 1 position to the right. A *value* of n moves the current location n positions to the right.
Cursor Backward <ESC>[*value*D	Move current location to left. *Value* specifies the distance moved; the default is 1. If *value* is 0 or 1, the current location is moved 1 position to the left. If *value* is n, the current location is moved n positions to the left.
Cursor Position <ESC>[*linenum;columnnum*H	Move current location to screen position specified by *line-num* and *column-num*. The default *values* for *line-num* and *column-num* are 1. If no parameters are present, GIGI moves the cursor to the home position.
Horizontal and Vertical Position <ESC>[*line-loc;column-loc*f	Move current location to position specified by *line-loc* and *col-loc*. A value of either 0 or 1 causes current location to move to first line or column in the display, respectively. When no line or column location is entered, the current location moves to location [1,1].
Index <ESC>D	Move current location down 1 line without changing the column position. If the current location is at the bottom margin and scrolling is enabled, a scroll up is performed.
Next Line <ESC>E	Move current position to the first position on the next line downward. If the current location is at the bottom margin and scrolling is enabled, a scroll up is performed.
Reverse Index <ESC>M	Move current location to same horizontal position on preceding line. If current position is at top margin and scrolling is enabled, a scroll down is performed.

Erasing Screen Areas

Erase to End of Screen <ESC>[0J or <ESC>[J	Erase from the cursor to end of the screen.
Erase from Top of Screen to Cursor <ESC>[1J	
Erase Entire Screen <ESC>[2J	Erase entire screen. All lines are erased, changed to single-width, and cursor does not move.
Erase From Cursor to End of Line <ESC>[0K or <ESC>[K	
Erase from Start of Line to Cursor <ESC>[1K	
Erase Entire Line <ESC>[2K	

Reports

Certain escape sequences provide for communications between the terminal and the host. For example, the host can transmit a sequence that requests the terminal to identify its type. The terminal then transmits a report to the host. This report is the response to the host request.

Cursor Position Report <ESC>[6n	When transmitted by the host, this sequence directs GIGI to report the cursor position to the host. GIGI sends the response in the form: <ESC>[P*line-num*;P*col-num*R where *line-num* and *col-num* specify the current line and column numbers. If there are no *values* specified for *line-num* or *col-num* or if they are 0, *line-num* and *col-num* are set to location [1,1].
Device Identification Report <ESC>[c or <ESC>[0c	When transmitted from the host, this sequence requests GIGI to send a device attribute control sequence to identify itself. GIGI sends the response in the form: <ESC>[?5;0c or <ESC>[?5c This response indicates that this is a GIGI terminal.

Device Status Report
<ESC>[*value*n

Reports the general status of the device sending the control sequence (parameter *value*s 0 and 3) and/or requests a status report from the receiving device (parameter *value*s 5 and 6), according to the parameter(s).

Parameter	Meaning
0 or 3	Ready, no malfunctions detected (default); response from VK100.
5	Please report status (using a control sequence); command from host.
6	Report current location (using a control sequence); command from host.

Control sequence with a *value* of 0 or 3 is sent by the VK100 as a response to a requesting control sequence (parameter *value* of 5).

Escape Sequences for Selecting Character Sets

ANSI conventions support the shift-in and shift-out functions for two character sets residing in terminal memory. By programming the ASCII control characters SI (shift-in) and SO (shift-out), you can alternate between character sets used by a terminal. When the terminal receives the SI control character, it activates the G0 character set; when the terminal receives the SO control character, it activates the G1 character set.

Because GIGI supports three alternate character sets, you can select any of the character sets listed below to be the G0 and G1 sets. For example, you could select the US ASCII set for G0 set and alternate character set 3, programmed for APL characters, to be the G1 set. The process for setting the G0 and G1 character sets is:

- Load appropriate characters into GIGI's character sets (use the ReGIS L (load) command, described in Part 2, *GIGI Graphics*).
- Map the G0 and G1 character sets to the escape sequences listed below:

	G0 Set	G1 Set
UK Character Set	<ESC>(A	<ESC>)A
US ASCII	<ESC>(B	<ESC>)B
Alternate Character Set 1	<ESC>(0	<ESC>)0
Alternate Character Set 2	<ESC>(1	<ESC>)1
Alternate Character Set 3	<ESC>(2	<ESC>)2

- Transmit the SI and SO control sequences from host programs to shift into and out of character sets.

  Alternate sets 1, 2, and 3 are the same sets referenced in ReGIS graphics mode. The native ASCII character set is preloaded with either the UK or US ASCII character set on terminal reset or power-up, depending on the setting of the UK setup parameter.

Escape Sequences for Select ANSI Graphics Renditions

Invoke Graphic Rendition
<ESC>[*value...*m

Invokes the graphic rendition specified by *value*. All following characters transmitted to the VK100 are rendered according to the *value* until the next graphics rendition is transmitted.

Parameter	Meaning
0 or None	All Attributes Off
2	Half Bright (or Green)
4	!Underscore On
5	Blink On
7	Reverse Video On
30	Black Writing
31	Red Writing
32	Green Writing
33	Yellow Writing
34	Blue Writing
35	Magenta Writing
36	Cyan Writing
37	White Writing
40	Black Screen
41	Red Screen
42	Green Screen
43	Yellow Screen
44	Blue Screen
45	Magenta Screen
46	Cyan Screen
47	White Screen

Other Escape Sequences

Reset to Initial State <ESC>c	Resets the VK100 to its initial state, but does not change any setup parameters.
Hardcopy <ESC>#7	This causes the screen to cease updating and freeze while the hardcopy output is enabled. The screen will resume normal operation when the hardcopy has been completed. (DEC Provate.)
Screen Alignment Display <ESC>#8	This command fills the entire screen area with a cross-hatch pattern for screen focus and alignment.

PROGRAMMING SET-UPS

You can set set-up parameters from programs by coding those programs to send appropriate Device Control Strings or escape sequences to GIGI. GIGI intercepts these strings or sequences and sets the appropriate set-up *value*. Programs that set set-ups can either execute in the host computer and send control sequences downline or they can reside in GIGI and issue control sequences from GIGI BASIC.

Using Device Control Strings to Set Set-ups

As described above, Device Control Strings are the mechanism GIGI uses to set various GIGI-specific functions. You can use the Device control String to set set-ups in the form:

<ESC>Pr *mnemonic value* <ESC>\

<ESC>Pr
Directs GIGI to set a set-up.

mnemonic
Specifies the mnemonic to be set.

value
specifies the *value* to set.

The trailing <ESC>\ terminates the control string.

No other characters (including spaces) are allowed in the control string; the spaces in the format are for readability. The following example sets the receive speed set-up (RS) to 4800 baud when you transmit the DCS:

<ESC>PrRS5<ESC>\

You can set all set-ups using the control string and the control string terminator. Refer to Part I, *Getting Started*, for a list of all GIGI set-ups and their functions.

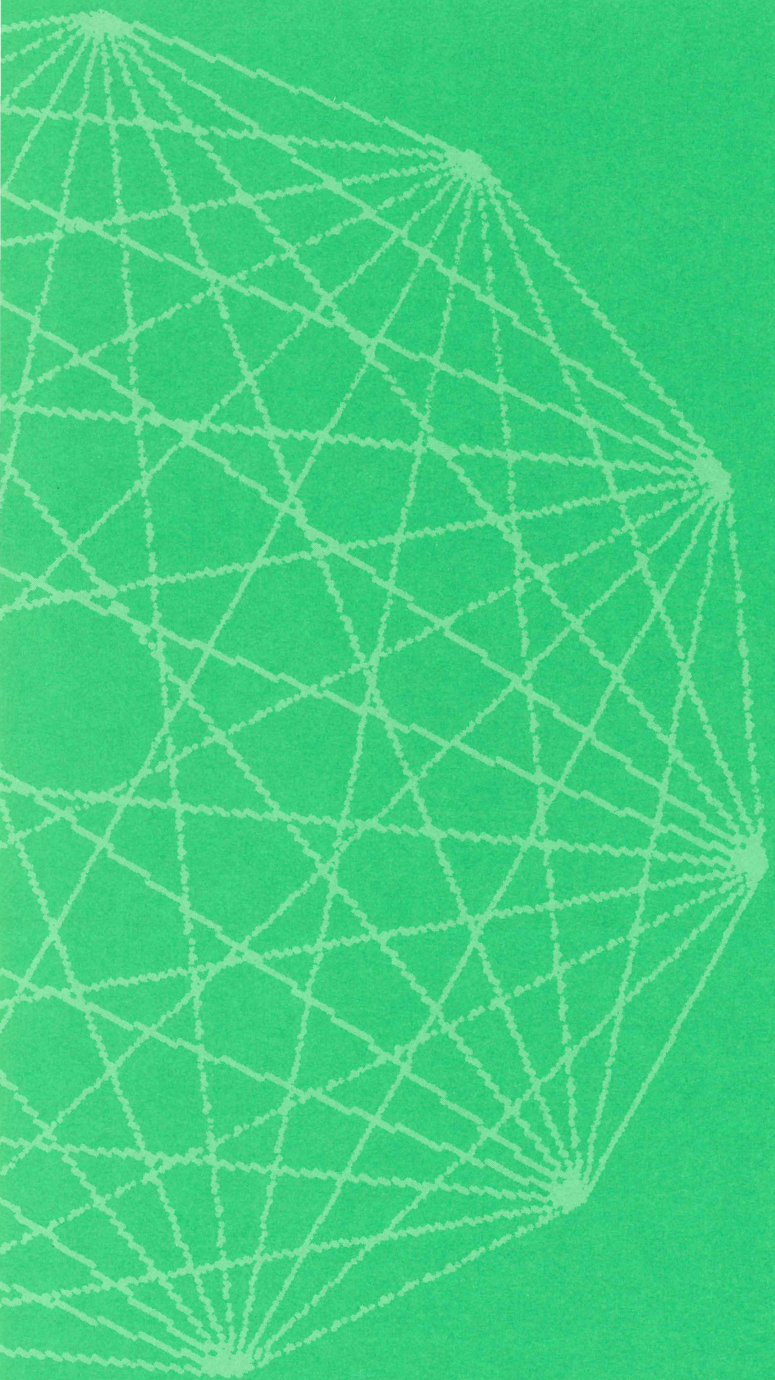
Using Escape Sequences to Set Setups

Certain set-ups can be changed by sending a special escape sequence to GIGI. These escape sequences are valid only in ANSI mode (Set-up TM1). Table 16-1 lists the set-ups that can be changed by means of escape sequences.

Table 16-1. Control Strings and Escape Sequences for Changing Set-ups

Set-up value	Meaning	Escape Sequence
AH0	Autohardcopy Off	<ESC>[?24l
AH1	Autohardcopy Enabled	<ESC>[?24h
AW0	Auto-wraparound Off	<ESC>[?7l
AW1	Auto-wraparound On	<ESC>[?7h
BA0	GIGI BASIC Off *	<ESC>[?21l
BA0	GIGI BASIC Off *	<ESC>[?22l
BA1	GIGI BASIC Enabled for Keyboard Input/Screen Output	<ESC>[?21h
BA2	GIGI BASIC Enabled for Host Control	<ESC>[?22h
CK0	Cursor Keys Normal	<ESC>[?1l
CK1	Cursor Keys Application	<ESC>[?1h
IL0	Interlace Off	<ESC>[?9l
IL1	Interlace On	<ESC>[?9h
KR0	Keyboard Autorepeat Off	<ESC>[?8l
KR1	Keyboard Autorepeat On	<ESC>[?8h
KP0	Application Keypad Off	<ESC>>
KP1	Application Keypad On	<ESC>=
NL0	New Line Off	<ESC>[2l
NL1	New Line On	<ESC>[2h
OS0	Overstrike Off	<ESC>[?20l
OS1	Overstrike On	<ESC>[?20h
PK0	Programmed Keypad Off	<ESC>[?23l
PK1	Programmed Keypad On	<ESC>[?23h
RV0	Normal Video	<ESC>[?5l
RV1	Reverse Video	<ESC>[?5h
TM0	Terminal Mode VT52	<ESC>[?2l
TM1	Terminal Mode ANSI	<ESC>[?2h
SM1	Jump Scroll	<ESC>[?4l
SM2	Smooth Scroll	<ESC>[?4h
UK0	Use US ASCII	<ESC>(A or)B
UK1	Use UK ASCII	<ESC>(A or)A

* Programming GIGI BASIC Mode: Escape sequences force termination of programs GIGI BASIC is executing; setting BA0 manually suspends execution of GIGI BASIC programs.



Appendices and Glossary

Appendix A
GIGI Key Codes

This appendix lists the keycodes GIGI generates. Most keys on GIGI's keyboard can be used in conjunction with the left and right SHIFT keys to generate two different codes. One set of codes is generated if neither SHIFT key is pressed. The other set of codes is generated if either or both of the SHIFT keys are pressed while the keys are pressed. Also, GIGI generates control codes when you press certain alphabetic keys at the same time you press the <CTRL> key. These codes are also listed.

Table A-1. Alphabetic Key Codes Transmitted by Main Keypad Keys

Key	Upper-case Code (Octal)	Lower-case Code (Octal)
A	101	141
B	102	142
C	103	143
D	104	144
E	105	145
F	106	146
G	107	147
H	110	150
I	111	151
J	112	152
K	113	153
L	114	154
M	115	155
N	116	156
O	117	157
P	120	160
Q	121	161
R	122	162
S	123	163
T	124	164
U	125	165
V	126	166
W	127	167
X	130	170
Y	131	171
Z	132	172

Table A-2. Alphabetic Key Codes Transmitted by Main Keypad Keys

Key	Lower-case Code (Octal)	Upper-case Code (Octal)
1	061	041 !
2	062	100 @
3	063	043 # or &
4	064	044 $
5	065	045 %
6	066	136 \
7	067	046 &
8	070	052 *
9	071	050 (
0	060	051)
—	055	137 _
=	075	053 +
[	133	173 {
;	073	072 :
'	047	052 "
,	054	074 <
.	056	076 >
/	057	077 ?
\	134	174]
`	140	176 ~
]	135	175 }

Table A-3. Codes Generated by Auxiliary Keypad

Key	Numeric Mode	Application Mode (ANSI)	(VT52)
0	0	<ESC>Op	<ESC>?p
1	1	<ESC>Oq	<ESC>?q
2	2	<ESC>Or	<ESC>?r
3	3	<ESC>Os	<ESC>?s
4	4	<ESC>Ot	<ESC>?t
5	5	<ESC>Ou	<ESC>?u
6	6	<ESC>Ov	<ESC>?v
7	7	<ESC>Ow	<ESC>?w
8	8	<ESC>Ox	<ESC>?x
9	9	<ESC>Oy	<ESC>?y
		<ESC>Om	<ESC>?m
,	,	<ESC>Ol	<ESC>?l
.	.	<ESC>On	<ESC>?n
ENTER	Same as RETURN	<ESC>OM	<ESC>?M
PF1	Same as in	<ESC>OP	<ESC>P
PF2	application	<ESC>OQ	<ESC>Q
PF3	mode	<ESC>OR	<ESC>R
PF4		<ESC>OS	<ESC>S

Table A-4. Keyboard-Generated Control Character Codes

Key Pressed with CTRL	Octal Code Transmitted	Mnemonic
SPACE	000	<NUL>
A	001	<SOH>
B	002	<STX>
C	003	<ETX>
D	004	<EOT>
E	005	<ENQ>
F	006	<ACK>
G	007	<BELL>
H	010	<BS>
I	011	<HT>
J	012	<LF>
K	013	<VT>
L	014	<FF>
M	015	<CR>
N	016	<SO>
O	017	<SI>
P	020	<DLE>
Q	021	<DC1> or <XON>
R	022	<DC2>
S	023	<DC3> or <XOFF>
T	024	<DC4>
U	025	<NAK>
V	026	<SYN>
W	027	<ETB>
X	030	<CAN>
Y	031	<EM>
Z	032	<SUB>
[	033	<ESC>
\	034	<FS>
]	035	<GS>
~	036	<RS>
?	037	<US>

Table A-5. Cursor Control Key Codes

Cursor Key	ANSI Mode	Mode	Application Mode
↑	<ESC>[A	<ESC>A	<ESC>OA
↓	<ESC>[B	<ESC>B	<ESC>OB
→	<ESC>[C	<ESC>C	<ESC>OC
←	<ESC>[D	<ESC>D	<ESC>OD

Table A-6. Control Character Set

GIGI Display	Octal Equivalent	Control Mnemonic	GIGI Display	Octal Equivalent	Control Mnemonic
^@	000	NUL	^P	020	DLE
^A	001	SOH	^Q	021	XON
^B	002	STX	^R	022	XOFF
^C	003	ETX	^S	023	DC3
^D	004	EOT	^T	024	DC4
^E	005	ENQ	^U	025	NAK
^F	006	ACK	^V	026	SYN
^G	007	BEL	^W	027	ETB
^H	010	BS	^X	030	CAN
^I	011	HT	^Y	031	EM
^J	012	LF	^Z	032	SUB
^K	013	VT	^[	033	ESC
^L	014	FF	^\	034	FS
^M	015	CR	^]	035	GS
^N	016	SO	^^	036	RS
^O	017	SI	^_	037	US

Appendix B
GIGI BASIC Commands and Statements

Statement Format	Usage
AUTO {*line number*{,*increment*}}	Generates a line number automatically after every carriage return.
CLEAR {,,*expression*}	Sets all numeric variables to zero and all string variables to null; and, optionally, sets the amount of stack space.
CONT	Continues program execution after a <CTRL>C has been typed, or a STOP or END statement has been executed.
{R}CTRL*char* CTRLC	Disables the checking for <CTRL>C for the program.
CTRLO	Disables the checking for <CTRL>O for the running program.
RCTRLC	Re-enables the checking for <CTRL>C.
RCTRLO	Re-enables the checking for <CTRL>O.
DATA *list of constants*	Stores the numeric and string constants that are accessed by the program's READ statement{s}.
DEF FN *name* {(*parameter*)} = *function definition*	Defines and names a function that is written by the user.
DELETE{*line number*}{-*line number*}	Deletes program lines.
DIM *list of subscripted variables*	Specifies the maximum values for array variable subscripts and allocates storage accordingly.
{NO}ECHO {#N}	Echoes of input to the implied or expressed data channel.
NOECHO	Disables echoing of input to the implied or expressed data channel.
EDIT *line number*	Enters Edit Mode at the specified line.
END	Terminates program execution and returns to command level.
ERASE *list of array variables*	Eliminates arrays from a program.
ERROR *integer expression*	Simulates the occurrence of a GIGI BASIC error, or allows error codes to be defined by the user.

Statement Format	Usage
FOR *variable* = *x* TO *y* [STEP *z*] . . . NEXT {*variable*}{,*variable...*}	Allows a series of instructions to be performed in a loop a given number of times.
GOSUB *line number* . . . RETURN	Branches to and returns from a subroutine.
GOTO *line number*	Branches unconditionally out of the normal program sequence to a specified line number.
HOST	Terminates interaction with GIGI BASIC and restores interaction with the host computer.
IF *expression* THEN *statement(s) line number* {ELSE *statement(s)* | *line number*} IF *expression* GOTO *line number* {ELSE *statement(s)* | *line number*}	Makes a decision regarding program flow based on the result returned by an expression.
INPUT{#N,}{;}{"prompt string";}*list of variables*	Allows input from the terminal or specified channel during program execution.
(LET) *variable* = *expression*	Assigns the value of an expression to a variable.
LINPUT{#N,}{;}{"prompt string";}*string variable*	Input an entire line (up to 254 characters) to a string variable, without the use of delimiters.
LIST {#n,} {*line number*{-{*line number*}}}	Lists all or part of the program currently in memory at the terminal.
MID$(*string exp1*,*n*{,*m*}) = *string exp2*	Replaces a portion of one string with another string.
NEW	Deletes the program currently in memory and clears all variables.
OLD *string constant or expression*	Requests a copy of a BASIC program from the host computer.
ON ERROR GOTO *line number*	Enables error trapping and specifies the first line of the error handling subroutines.
ON *expression* GOTO *list of line numbers* ON *expression* GOSUB *line number*	Branches to one of several specified line numbers, depending on the value returned when an expression is evaluated.

Statement Format	Usage
OPTION BASE *n*	Declares the minimum value for array subscripts.
OUT I,J	Sends a byte to a machine output port.
PRINT {#N,} {*list of expressions*}	Output data at the terminal or specified channel.
RANDOMIZE {*expression*}	Resets the random number generator.
READ *list of variables*	Reads values from a DATA statement and assigns them to variables.
REM *remark*	Allows explanatory remarks to be inserted in a program.
RESTORE {*line number*}	Allows DATA statements to be reread from a specified point.
RESUME	Continues program execution after an error recovery procedure has been performed.
RUN {*line number*}	Executes the program currently in memory.
SAVE *string constant or expression*	Sends a copy of the GIGI BASIC program to the host computer.
STOP	Terminates program execution and returns to command level.
SWAP *variable,variable*	Exchanges the values of two variables.
TRON TROFF	Traces the execution of program statements. Disables tracing the execuction of program statements.
WAIT *port number,* I{,J}	Suspends program execution while monitoring the status of a machine input port.
WHILE *expression* . . . WEND	Executes a series of statements in a loop as long as a given condition is true.
WIDTH *integer expression*	Sets the printed line width in number of characters for the terminal or printer.

Statement Format	Usage
FUNCTIONS	
ABS(X)	Returns the absolute value of the numeric expression.
ASC(X$)	Returns a numerical value that is the ASCII code of the first character of the specified string expression.
ATN(X)	Returns the arctangent of the specified numeric expression in radians.
CHR$(I)	Returns a string whose one element has ASCII code for the specified integer expression.
COS(X)	Returns the cosine of the specified numeric expression in radians.
ESC$	Returns a string of length one containing the ESCAPE character.
EXP(X)	Returns e to the power of the specified numeric expression.
FRE(0) FRE(X$)	Returns the number of bytes in memory not being used by GIGI BASIC.
GOFF$	Generates a string of length 2 that contains the "graphics off" escape sequence (i.e., ESCAPE followed by a backslash) that causes GIGI to leave graphic mode. To take effect, the escape sequence must be sent to GIGI via a PRINT statement.
GON$	Generates a 3-character string that contains the "graphics on" control string (the ESCAPE string followed by Pp) that causes GIGI to enter graphics mode if sent to GIGI in a PRINT statement.
HEX$(X)	Returns a string which represents the hexadecimal value of the decimal argument. The specified decimal argument is rounded to an integer before it is evaluated.
INP(I)	Returns the byte read from the specified port.
INKEY${#N}{W}	Returns a 1-character string, read from the terminal or from the terminal or from the specified channel.
INSTR({I,}X$,Y$)	Searches for the first occurrence of a string expression in another string expression and returns the position at which the match is found.
INT(X)	Returns the largest integer less than or equal to the specified numeric expression.
LEFT$(X$,I)	Returns a string comprised of the specified number of characters of the string expression, beginning from the left.
LEN(X$)	Returns the number of characters in the specified string expression.

Statement Format	Usage
LOG(X)	Returns the natural logarithm of the specified numeric expression (the numeric expression must have a value greater than 0).
MID$(X$,I{,J})	Returns a string of the specified length from the specified string, beginning at the specified character position.
OCT$(X)	Returns a string which represents the octal value of the decimal argument. (The decimal argument is rounded to an integer before it is evaluated).
POS(I)	Returns the current cursor position.
RIGHT$(X$,I)	Returns a string of the specified number of characters of the string expression, starting from the right.
RND{(X)}	Returns a random number between 0 an 1.
SGN(X)	Returns a value of 0 if the specified numeric expression is equal to 0; if the value of the expression is greater than 0, a 1 is returned; if the value of the expression is less than 0, a -1 is returned.
SIN(X)	Returns the sine of the specified numeric expression in radians.
SPACE$(X)	Returns a string of spaces of the specified length (length is rounded to an integer).
SPC(I)	Prints the specified numer of blanks on terminal (only valid for a PRINT or LPRINT statement).
SQR(X)	Returns the square root of the specified numeric expression. The value of the expression must be greater than or equal to 0.
STR$(X)	Returns a string representation of the specified numeric expression.
STRING$(I,J)	Returns a string of the length specified whose characters all have ASCII code of the specified integer or of the first character of the specified string.
TAB(I)	Spaces to the specified terminal position.
TAN(X)	Returns the tangent of the specified numeric expression in radians.
VAL(X$)	Returns the numerical value of the specified string expression.

Appendix C
Octal and Decimal ASCII Key Codes

Char	Oct	Dec	Char	Oct	Dec	Char	Oct	Dec	Char	Oct	Dec
<NUL>	000	000	space	040	032	@	100	064	'	140	096
<SOH>	001	001	!	041	033	A	101	065	a	141	097
<STX>	002	002	"	042	034	B	102	066	b	142	098
<ETX>	003	003	#	043	035	C	103	067	c	143	099
<EOT>	004	004	$	044	036	D	104	068	d	144	100
<ENQ>	005	005	%	045	037	E	105	069	e	145	101
<ACK>	006	006	&	046	038	F	106	070	f	146	102
<BELL>	007	007	'	047	039	G	107	071	g	147	103
<BS>	010	008	(	050	040	H	110	072	h	150	104
<HT>	011	009	)	051	041	I	111	073	i	151	105
<LF>	012	010	*	052	042	J	112	074	j	152	106
<VT>	013	011	+	053	043	K	113	075	k	153	107
<FF>	014	012	,	054	044	L	114	076	l	154	108
<CR>	015	013	—	055	045	M	115	077	m	155	109
<SO>	016	014	.	056	046	N	116	078	n	156	110
<SI>	017	015	/	057	047	O	117	079	o	157	111
<DLE>	020	016	0	060	048	P	120	080	p	160	112
<DC1>	021	017	1	061	049	Q	121	081	q	161	113
<DC2>	022	018	2	062	050	R	122	082	r	162	114
<DC3>	023	019	3	063	051	S	123	083	s	163	115
<DC4>	024	020	4	064	052	T	124	084	t	164	116
<NAK>	025	021	5	065	053	U	125	085	u	165	117
<SYN>	026	022	6	066	054	V	126	086	v	166	118
<ETB>	027	023	7	067	055	W	127	087	w	167	119
<CAN>	030	024	8	070	056	X	130	088	x	170	120
<EX>	031	025	9	071	057	Y	131	089	y	171	121
<SUB>	032	026	:	072	058	Z	132	090	z	172	122
<ESC>	033	027	;	073	059	[	133	091	{	173	123
<FS>	034	028	<	074	060	\	134	092	]	174	124
<GS>	035	029	=	075	061	]	135	093	}	175	125
<RS>	036	030	>	076	062	^	136	094	–	176	126
<US>	037	031	?	077	063	_	137	095	<DEL>	177	127

GIGI GLOSSARY

A

absolute location: an unsigned coordinate pair that specifies a location based on the origin location.

alternate character set: one of the three user-definable character sets in GIGI; character sets 1, 2, and 3.

arrow keys: keys at the right of the main keypad marked with arrows and used, in general, to move the cursor.

attributes: a characteristic or distinctive feature; for GIGI there are screen, writing, and text attributes.

auto-repeat: the terminal feature which causes the continuous transmission of the character for as long as you press the key for that character.

auto-wraparound: the terminal feature which continues the current line on the next line when the end of the current line is reached.

auxiliary keypad: the section of the keyboard to the right of the main keypay consisting of the numeric pad and program function keys; often programmed for application-specific purposes.

B

BASIC interpreter: the firmware in GIGI which processes GIGI BASIC statements.

BASIC mode: operating mode in which all terminal functions are under the control of the GIGI BASIC interpreter.

bracketed pair: a pair of values enclosed in brackets; used in ReGIS command strings.

brightness: monitor control used to adjust the intensity of the screen.

C

character: an 8-bit ASCII code; displayable characters are represented by an eight-by-ten dot pattern that is the basic unit in each of GIGI's four character sets.

character cell: an eight-by-ten cell which contains a character pattern.

character cell origin: the top left corner of the character cell.

character file: a file which contains the definition of a character or group of characters.

character pattern: the dot pattern contained in a character cell.

character set: a group of 128 characters; for GIGI's alternate characters sets, 95 characters are displayable.

character size: the attributes defining the width and height of a character.

closed curve sequence: a series of locations that GIGI uses to interpolate a curve whose end points meet.

color: an attribute which defines the screen or writing color.

command keyletter: a letter which directs the ReGIS interpreter to perform a graphics operation.

complement writing: writing mode in which GIGI complements the existing image as new images are written to the screen.

computer assisted instruction (CAI): use of computer to augment the individual instruction process by providing the student with programmed sequences of instruction under computer control, permitting students to progress at their own rate.

coordinate pair: an x-coordinate and a y-coordinate which together define a location.

coordinate: a number used to specify the location or point on a line.

current character set: the most-recently specified character set.

current location: a location GIGI maintains internally which is a pointer to the location last moved to or drawn to.

cursor keys: keys used to move the graphics or text cursor; generally refers to the arrow keys on the main keypad.

D

default: a value assumed when no specific choice is given by the user or a program.

down-line load: the process of transferring a program from one system to another via a communication line.

dot: see pixel

drawing: see writing.

E

echo: retransmit to the sender that which is sent.

erase writing: writing operation which removes previously-drawn objects.

G

GIGI: General Imaging Generator and Interpreter

graphic cursor: a diamond-shaped cursor displayed only when GIGI is in graphic mode.

graphic mode: terminal operating mode in which the ReGIS interpreter is enabled.

graphic text: text displayed in graphics mode.

gray scale: 8 levels of intensity GIGI uses on black and white monitors.

H

host system: computer system (hardware and software) to which GIGI is connected.

I

image: all objects displayed on the screen.

image memory: a bit map GIGI uses to maintain and display images on the screen.

interlace: display of every other horizontal line of pixels; helps reduce flicker on low resolution monitor screens.

italic: character slant.

K

keyboard: GIGI's main keypad and the auxiliary keypad.

L

label: an identifier used by the Graphics Editor to distinguish an object.

line pattern: an eight-dot pattern GIGI uses to write with

location: a point defined by a coordinate pair.

locator cursor: the cursor displayed when in locator mode; cursor is a large cross-hair.

logical screen coordinates: user-defined screen coordinates

M

macrograph: a named string which can be recalled and which may consist of ReGIS commands.

main keypad: that portion of the keyboard consisting of the alphanumeric characters.

mnemonic: an abbreviation or acronym that is easy to remember.

monitor: a separate video device containing the cathode ray tube (CRT) GIGI uses to display screen images.

mosaic: a multi-character image usually created with characters from alternate character sets.

multiplication factor: number used to multiply the pixels in either a writing pattern or a text cell before displaying the pattern or cell.

N

native character set: the 128 ASCII character set; for GIGI 95 of these characters are displayable.

negative writing: the reversed interpretation of the dot pattern GIGI uses for writing.

O

offset: a distance from a given location.

offset direction: a numbered direction which directs GIGI to write in a specific direction in relationship to the current location.

open curve sequence: a series of points that GIGI uses to interpolate a curve whose end points do not meet.

overlay writing: writing mode in which GIGI overlays an existing image with new images written to the screen.

P

picture: any combination of drawings and text displayed on GIGI's screen.

pixel: picture element; the smallest displayable unit on the video monitor screen.

program function keys: top four keys on the auxilliary keypad, labelled PF1, PF2, PF3, and PF4.

R

relative location: a point on the screen measured from the the current location rather than the screen origin.

ReGIS: Remote Graphics Instruction Set.

replace writing: a writing mode in which GIGI replaces existing images as new images are written to the screen.

reset: return to a known default condition.

reverse video: the reverse interpretation of the screen and image bits in video memory; screen color becomes drawing color and drawing color becomes screen color.

S

screen: that portion of the video monitor on which images are displayed.

scrolling: the continuous horizontal or vertical movement of objects to make room for new objects.

self-tests: internal tests of terminal hardware which GIGI performs.

set-up mode: GIGI operating mode in which set-up parameters can be changed.

set-up parameter: terminal characteristic that can be changed in set-up mode to adapt the terminal to the operating environment.

shade character: user-specifield character GIGI uses during a write operation with shading enabled.

shade reference line: Y-coordinate GIGI uses to delimit an area to be shaded during a writing operation.

shading: writing a shading pattern during a writing operation.

standalone: GIGI operating mode in which GIGI is not communicating with the host system.

T

temporary writing controls: writing controls which apply to execution of a single ReGIS command.

text cursor: block cursor GIGI displays when not in graphics mode.

text mode: terminal operating mode in which GIGI operates as an ANSI or VT52 terminal; graphics interpretation is disabled.

V

vector: a directed line.

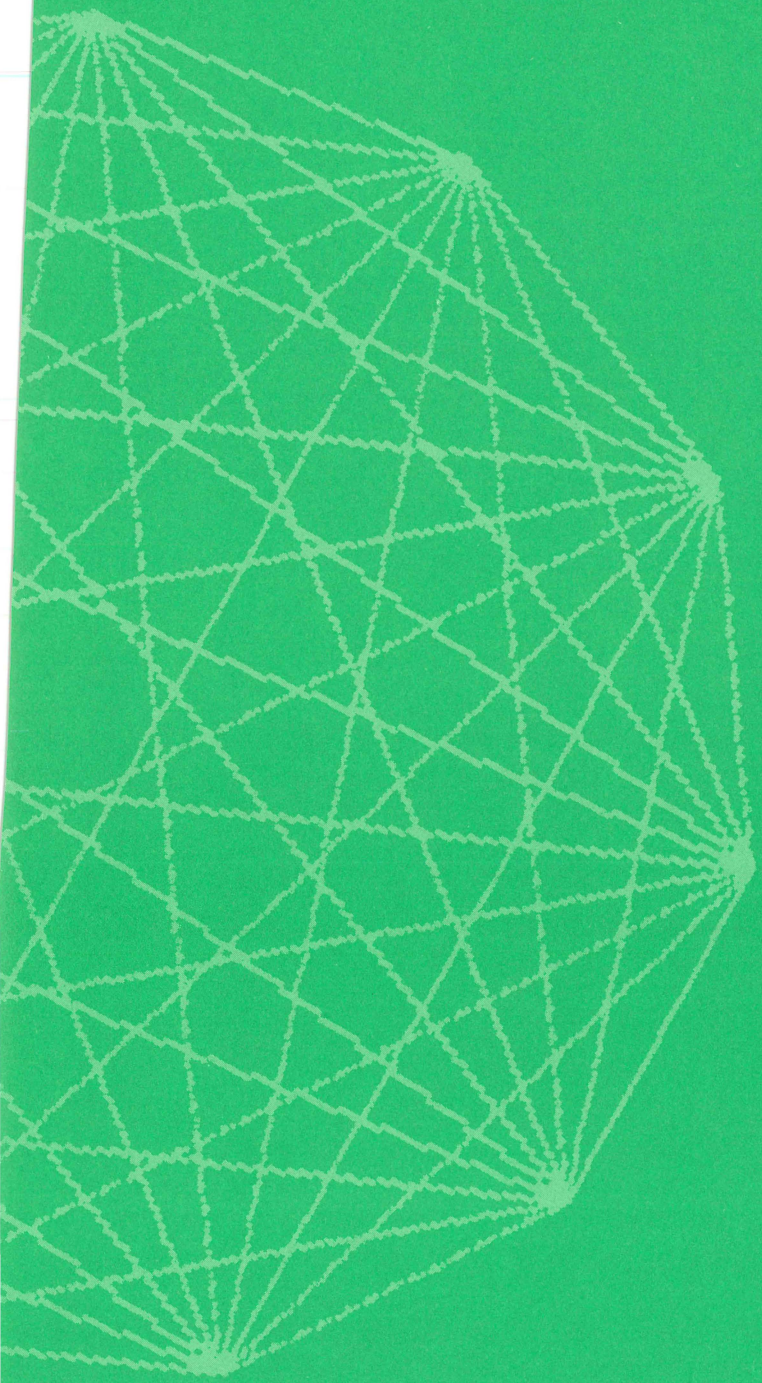
W

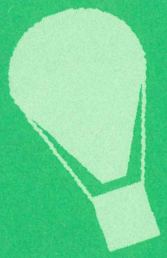
writing: the operation GIGI performs to display lines or text on the screen.

X

XON/XOFF: transmit on/transmit off control characters used to synchronize the terminal buffer with the host system.

Index





INDEX

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

READER'S COMMENTS

This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. Problems with software should be reported on a Software Performance Report (SPR) form. If you require a written reply and are eligible to receive one under the SPR service, submit your comments on an SPR form.

- Did you find errors in this manual? If so, please specify by page.

- Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

- Please indicate the type of user/reader that you most nearly represent.

☐ Assembly language programmer
☐ Higher-level language programmer
☐ Occasional programmer
☐ User with little programming experience
☐ Student programmer
☐ Nonprogrammer interested in computer concepts and capabilities

Name Date

Organization

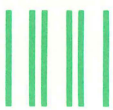
Street

City State Zip Code or Country

Please cut along this line.

Fold here----------

Do Not Tear — Fold Here and Staple----------



No Postage
Necessary
If Mailed In The
United States

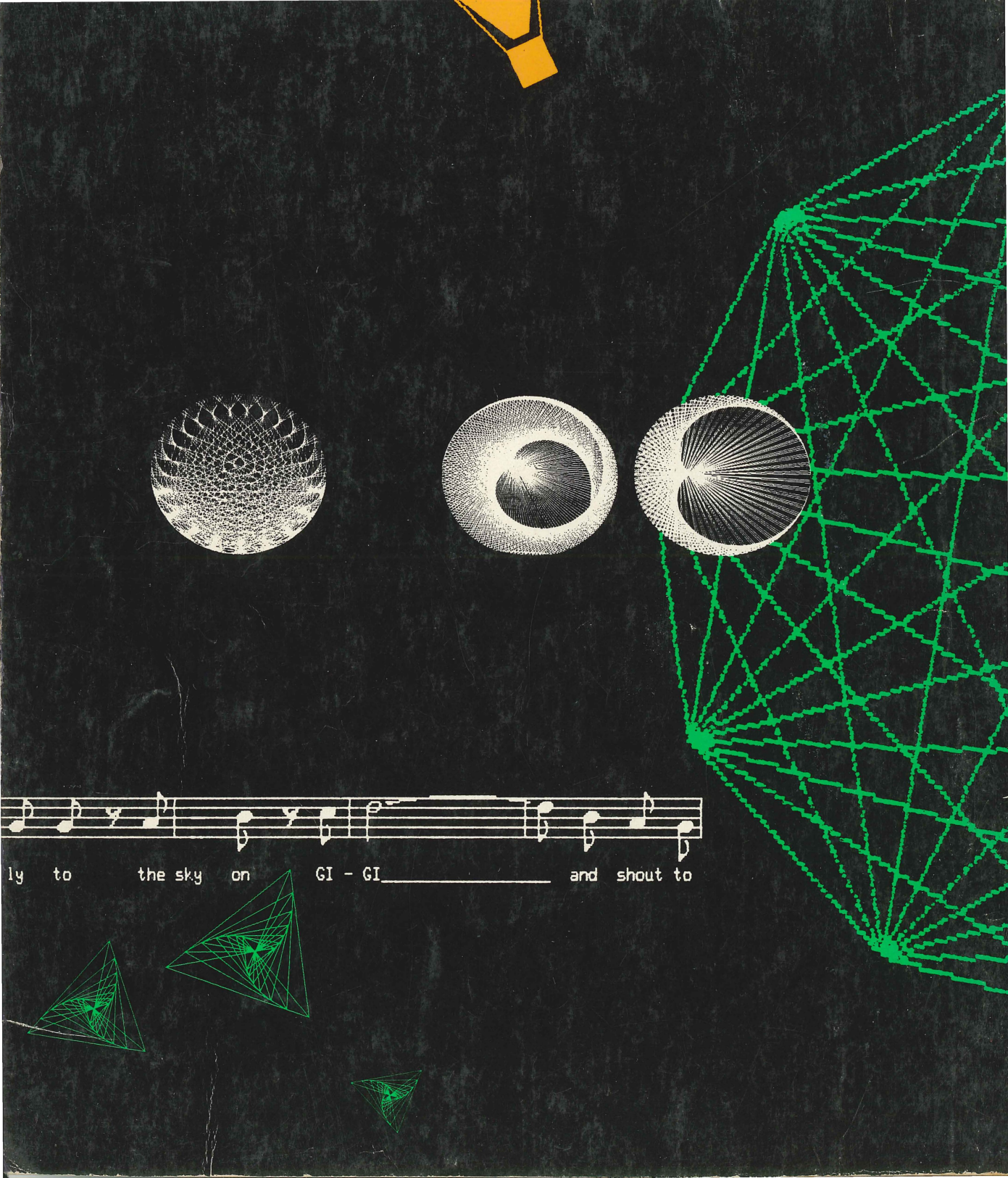
BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.152 MARLBOROUGH, MA 01752

Postage will be paid by:

digital

Education Computer Systems
ECS Engineering
26 Research Drive (EH/1)
Westboro, Massachusetts 01581


ly to the sky on GI - GI_____ and shout to